Tasks Sarika V

Task1

Task1

Top 10 notorious hackers in the internet:

1. Kevin Mitnik:

a. Now white hat hacker, then black hat with 5yrs of prison sentence.

b. He used cloned cellular phones to hide his location and, among other things, copied valuable proprietary software from some of the country's largest cellular telephone and computer companies. Mitnick also intercepted and **stole computer passwords, altered computer networks, and broke into and read private emails**. Stole **computer codes**

c. Social Engineering

d. Gray hat

2. Anonymous

a. Anonymous is a collective of online "**hacktivists**" whose origins can be traced to the early days of 4chan, an anonymous imageboard

b. Members would organize raids of online chatrooms to troll users, but with time, their antics became socially and politically motivated. Like cuber war against Russian for its invasion of Ukraine. Dozens of hackers with alleged ties to Anonymous have been arrested, but with no real leadership or structure, the group continues to launch cyberattacks on world governments and million-dollar corporations **in the name of social justice**.

c. Gray hat

3. Adrian Lamo

a. In 2001, 20-year-old Adrian Lamo used an unprotected content management tool at **Yahoo** to modify a Reuters article and add a fake quote attributed to former Attorney General John Ashcroft. Lamo often hacked systems and then notified both the press and his victims. In some cases, he'd help clean up the mess to improve their security. As Wired points out, however, Lamo took things too far in 2002, when he hacked **The New York Times' intranet**, added himself to the list of expert sources and began conducting research on high-profile public figures.

b. Lamo earned the moniker "The Homeless Hacker" because he preferred to wander the streets with little more than a backpack and often had no fixed address.

c. Grey hat

4. Albert Gonzalez

a. His start as the "troubled pack leader of computer nerds" at his Miami high school. He eventually became active on criminal commerce site Shadowcrew.com and was considered one of its best hackers and moderators. At 22, Gonzalez was arrested in New York for **debit card fraud** related to stealing data from millions of card accounts. To avoid jail time, he became an informant for the Secret Service, ultimately helping indict dozens of Shadowcrew members.

b. During his time as a paid informant, Gonzalez continued his in criminal activities. Along with a group of accomplices, Gonzalez stole more than 180 million payment card accounts from companies including OfficeMax, Dave and Buster's and Boston Market. Gonzalez's 2005 attack on US retailer TJX was the first serial data breach of credit information.

c. Using a basic **SQL injection**, this famous hacker and his team created back doors in several corporate networks, stealing an estimated $256 million from TJX alone. During his sentencing in 2015, the federal prosecutor called Gonzalez's human victimization "unparalleled."

d. Black hat

5. Mathewbevan and Richard pryce

a. Matthew Bevan and Richard Pryce are a team of British hackers who hacked into **multiple military networks** in 1996, including Griffiss Air Force Base, the Defense Information System Agency and the Korean Atomic Research Institute (KARI). Bevan (Kuji) and Pryce (Datastream Cowboy) have been accused of nearly starting a third world war after they dumped KARI research onto American military systems.

b. Bevan claims he was looking to prove a UFO conspiracy theory, and according to the BBC, his case bears resemblance to that of Gary McKinnon. Malicious intent or not, Bevan and Pryce demonstrated that even military networks are vulnerable.

c. Gray hat

6. Jeanson James Ancheta

a. Jeanson James Ancheta had no interest in hacking systems for credit card data or crashing networks to deliver social justice. Instead, Ancheta was curious about the **use of bots—**software-based robots that can infect and ultimately control computer systems. Using a series of large-scale "botnets," he was able to compromise more than 400,000 computers in 2005. According to Ars Technica, he then rented these machines out to advertising companies and was also paid to directly install bots or adware on specific systems. Ancheta was sentenced to 57 months in prison. This was the first time a hacker was sent to jail for the use of botnet technology.

b. Black hat

7. Michael Calce

a. In February 2000, 15-year-old Michael Calce, also known as "Mafiaboy," discovered how to take over **networks of university computers**. He used their combined resources to disrupt the number-one search engine at the time: Yahoo. Within one week, he'd also brought down Dell, eBay, CNN and Amazon using a **distributed-denial-of-service (DDoS) attack** that overwhelmed corporate servers and caused their websites to crash. Calce's wake-up call was perhaps the most jarring for cyber crime investors and internet proponents. If the biggest websites in the world—valued at over $1 billion—could be so easily sidelined, was any online data truly safe? It's not an exaggeration to say that the development of cyber crime legislation suddenly became a top government priority thanks to Calce's hack.
b. Black hat
8. Kevin Poulsen

a. In 1983, a 17-year-old Poulsen, using the alias Dark Dante, hacked into ARPANET, the **Pentagon's computer network**. Although he was quickly caught, the government decided not to prosecute Poulsen, who was a minor at the time. Instead, he was let off with a warning.
b. But still he hacked a federal computer and dug into files pertaining to the deposed president of the Philippines, Ferdinand Marcos. When discovered by authorities, Poulsen went underground. While he was on the run, Poulsen kept busy, **hacking government files and revealing secrets**. According to his own website, in 1990, he hacked a radio station contest and ensured that he was the 102nd caller, winning a brand new Porsche, a vacation, and $20,000.
c. Poulsen was soon arrested and barred from using a computer for three years. He has since converted to white hat hacking and journalism, writing about cyber security and web-related socio-political causes for Wired, The Daily Beast and his own blog Threat Level. Paulson also teamed with other leading hackers to work on various projects dedicated to social justice and freedom of information. Perhaps most notably, working with Adam Swartz and Jim Dolan to develop the open-source software SecureDrop, initially known as DeadDrop. Eventually, Poulsen turned over the platform, which enabled secure communication between journalists and sources, to the Freedom of Press Foundation.
d. Now white hack, back then before his arrest black hat
9. Jonathan james

a. Using the alias cOmrade, Jonathan James hacked several **companies**. According to the New York Times, what really earned James attention was his hack into the computers of the United States Department of Defense. Even more impressive was the fact that James was only 15 at the time. James admitted that he was partly inspired by the book *The Cuckoo's Egg*, which details the hunt for a computer hacker in the 1980s. His hacking allowed him to access over **3,000 messages from government employees, usernames, passwords and other sensitive data**.

b. James was arrested in 2000 and was sentenced to a six months house arrest and banned from recreational computer use. However, a probation violation caused him to serve six months in jail. Jonathan James became the **youngest person** to be convicted of violating cyber crime laws. In 2007, TJX, a department store, was hacked and many customer's private information were compromised. Despite a lack of evidence, authorities suspect that James may have been involved.

c. In 2008, James committed suicide by gunshot (psychological reasons, imp of mental health). According to the Daily Mail, his suicide note stated, "I have no faith in the 'justice' system. Perhaps my actions today, and this letter, will send a stronger message to the public. Either way, I have lost control over this situation, and this is my only way to regain control."

d. Black hat

10. Astra

a. Infiltrated French company with civil and military aviation subsidies, and stole and subsequently sold corporate secrets, including **information on weapons systems**. Dassault reported damages in excess of $361 million. Astra was arrested in Athens, Greece, in 2008.

Greek authorities have yet to release his name to the public but describe him as “a 58-year-old mathematician”, never publicly identified.

b. He had been hacking into the Dassault Group, for almost half a decade, stole cutting edge weapons technology software and data which he then sold to 250 individuals around the world, $360 million in damages. No one knows why his complete identity has never been revealed, but the word 'ASTRA' is a Sanskrit word for 'weapon'.

c. Black hat

Task2

Task 2

Vulnerabilities at each ports

1. FTP-Data

a. Use: Send and receive files from servers.

b. Outdated and insecure, no data encryption or authentication.

c. Vulnerability:

i. Brute-forcing passwords

ii. Anonymous authentication (it’s possible to log into FTP port with “anonymous” as the username and password)

iii. Cross-site scripting

iv. Directory traversal attacks

2. FTP

. Expose sensitive information and network credentials to an attacker when transmitting data across the network or the Internet.

i. doesn’t have encryption for data transfer or authentication, cross-site scripting, brute-forcing passwords, and directory traversal attacks.

3. SSH

. Use: to connect to servers, make changes, perform uploads and exit, either using tools or directly through the terminal

a. Vulnerabilities:

. Brute-force attack

i. enabling a remote attacker with access to the remote server where a user's SSH-agent is forwarded to load and unload shared libraries in /usr/lib* on the user's workstation.

4. TELNET

. Use: Lets users connect to remote devices and computers.

a. Superseded by SSH, but still used by some websites. Outdated and insecure

b. Vulnerabilities:

. Spoofing

i. Malware

ii. Credential brute-forcing

iii. Credential sniffing

5. SMTP

. Use: Receive and send mails

a. With no proper configuration and protection, if insecure is vulnerable

b. Vulnerabilities:

. Spoofing

i. Mail Spamming

6. DNS

. Use: zone transfers and maintaining coherence between the server and DNS database.

a. Vulnerabilities:

. DDos attack

i. DNS spoofing
ii. DNS tunneling (attackers manage remote servers and applications)
iii. DNS spoofing

7. TFTP

. Use: send and receive files between users and servers, no need for authentication
a. Faster but less secure
b. Vulnerabilities:
. DoSattack
i. execute arbitrary code via a long TFTP error packet
ii. access to files outside the restricted directory.

8. HTTP

. Use: By servers, most used
a. Vulnerabilities:
. Expose server components
i. Cross-site scripting
ii. SQL injections
iii. Cross-site request forgery
iv. DDoS attacks

9. POP3

. Use: Client mail protocol used by email clients to synchronize and download mail from remote mail servers
a. Vulnerabilities:
. Re-usable cleartext password.
i. No auditing of connections & attempts, thus subject to grinding.
ii. Buffer overflow problems.

10. NTP

. Use: allows the synchronization of system clocks (from desktops to servers). Having synchronized clocks is not only convenient but required for many distributed applications. Therefore the firewall policy must allow the NTP service if the time comes from an external server.
a. Vulnerabilities:
. denial of service (DoS) attack
i. NTP is vulnerable to MitM attacks. These attacks allow unauthorized users to intercept, read, and modify traffic sent between clients and servers. NTP is particularly susceptible to MitM attacks due to the reliance on a small set of servers and the algorithm used to choose a server with which to sync.

11. IMAP

. Use:
a. Vulnerabilities:
. Data interception, it transmits logins from the client to the server in plain text by default, meaning usernames and passwords are not encrypted.
i. Spoofing
ii. Password spraying attacks

12. HTTPS

. Vulnerabilities:
. Heart bleed (stealing the information protected)
i. CCS
ii. POODLE, SSL (CVE-2014-3566)
iii. Secure Renegotiation (CVE-2009-3555)
iv. CRIME, TLS (CVE-2012-4929)
v. BREACH (CVE-2013-3587)

Task3

Task 3

OWASP Vulnerabilities:
//cwe=common weakness enumeration

1. Broken Access Control
CWE: CWE-284 Improper Access Control
OWASP Category: A01:2021-Broken Access Control
Description: The product does not restrict or incorrectly restricts access to a resource from an unauthorized actor.
Business Impact: In case the product does not restrict access, there is a compromise in the data confidentiality and integrity. Unauthorized people can access and alter the data any way they want. Access of data by unauthorized means, the data can be made public and the result can be devastating, especially sensitive information like bank credentials and personal information of the user. Denial of Service can also be observed if the access control is incorrect. Arbitrary code can be easily executed and makes the product vulnerable.

2. Cryptographic Failures
CWE: CWE-327 Use of Broken or Risky Cryptographic Algorithm
OWASP Category: A02:2021-Cryptographic Failures
Description: The product uses a broken or risky cryptographic algorithm or protocol.
Business Impact: Encryption is performed so that data can be shared safe and secure over the network, even if accessible it should be unreadable which makes sure that the data is confidential. Cryptographic algorithms are the methods by which data is scrambled to prevent observation or influence by unauthorized actors. Using broken or risky cryptographic algorithms makes it easy for attackers to access the data easily even if encrypted. There is a threat to protection and confidentiality of the data. Data can be modified and spoofing is possible.

3.Injection
CWE: CWE-94 Improper Control of Generation of Code (‘Code Injection’)
OWASP Category: A03:2021-Injection
Description: The product constructs all or part of a code segment using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the syntax or behavior of the intended code segment.
Business Impact: Improper validation of inputs from the product entered by the user is vulnerable. Since these codes can have a direct impact on control flow or the configuration of the application, they are a greater threat to the service provided. This allows injection of control plane data into user-controlled data planes. Sending code through legitimate data channels can alter the execution. SQL injection and format string vulnerabilities are most observed methods used by attackers.

4. Insecure Design
CWE: CWE-657 Violation of Secure Design Principles
OWASP Category: A04:2021-Insecure Design
Description: The product violates well-established principles for secure design.
Business Impact: Insecure design violating or not following principles and guidelines can be very vulnerable. It can introduce threats or makes it easier for attackers to use the vulnerabilities during implementation. As the code is centered around the design, it becomes difficult and

requires a lot of resources to fix it. It can have flaws in security settings and configuration which makes it easier for attackers.

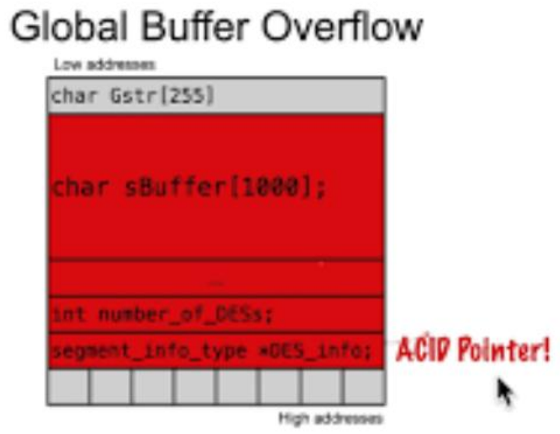
5. Security Misconfiguration
CWE: CWE-614 Sensitive Cookie in HTTPS session without ‘Secure’ attribute
OWASP Category: A05:2021-Security Misconfiguration
Description: The Secure attribute for sensitive cookies in HTTPS sessions is not set, which could cause the user agent to send those cookies in plaintext over an HTTP session.
Business Impact: Cookies contain sensitive information. When these are not secure, the information is sent in plain text. Unencrypted data can be easily intercepted and exploited. This is a threat to confidentiality of data. The attacker gains unauthorized access to data, system and network which makes the product untrustable and insecure.

Task4

Task4
Understanding web application attack

1. Out-of-bound write

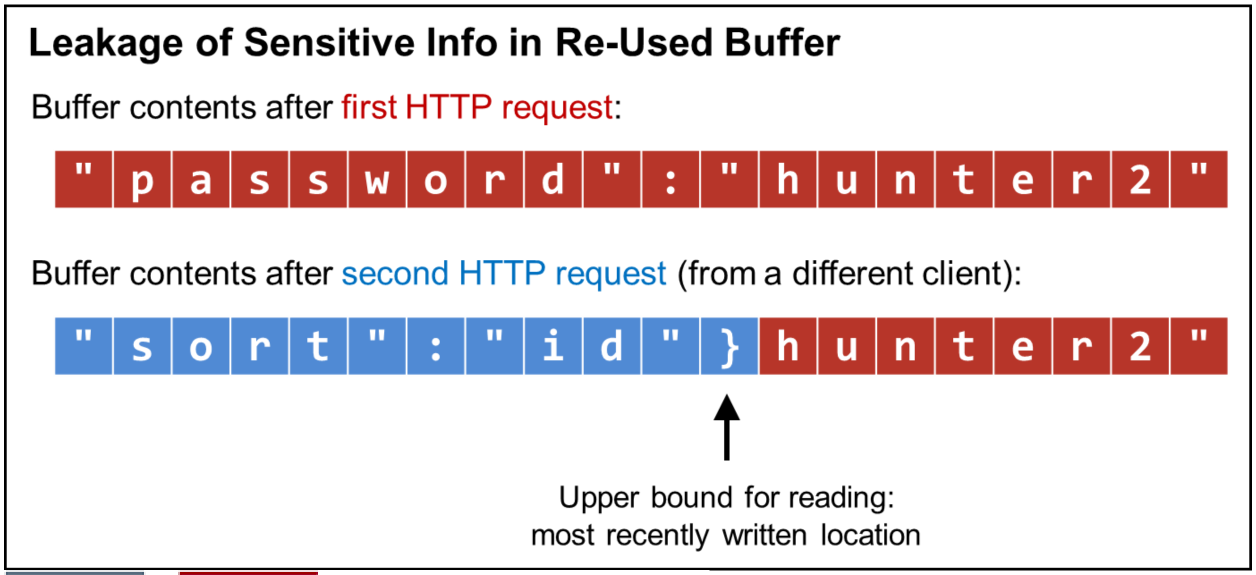
The product writes data past the end, or before the beginning, of the intended buffer. Typically, this can result in corruption of data, a crash, or code execution. The product may modify an index or perform pointer arithmetic that references a memory location that is outside of the boundaries of the buffer. A subsequent write operation then produces undefined or unexpected results.
Often used to describe the consequences of writing to memory outside the bounds of a buffer, or to memory that is invalid, when the root cause is something other than a sequential copy of excessive data from a fixed starting location. This may include issues such as incorrect pointer arithmetic, accessing invalid pointers due to incomplete initialization or memory release, etc.



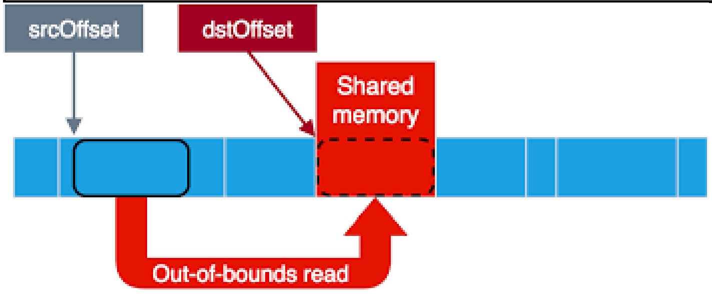
2. Out-of-bounds read

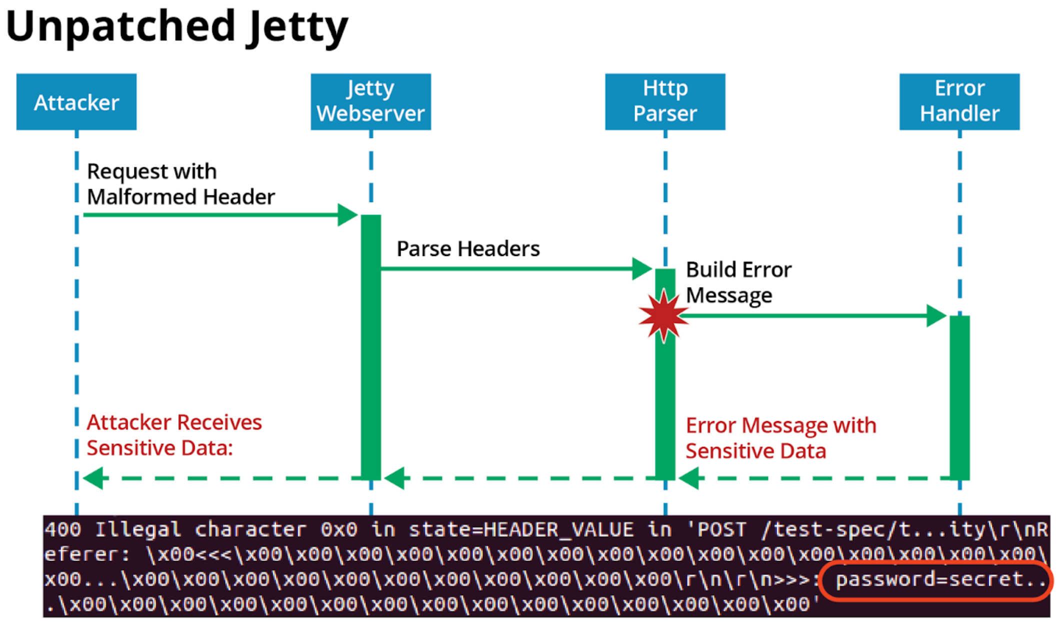
The product reads data past the end, or before the beginning, of the intended buffer.

Typically, this can allow attackers to read sensitive information from other memory locations or cause a crash. A crash can occur when the code reads a variable amount of data and assumes that a sentinel exists to stop the read operation, such as a NUL in a string. The expected sentinel might not be located in the out-of-bounds memory, causing excessive data to be read, leading to a segmentation fault or a buffer overflow. The product may modify an index or perform pointer arithmetic that references a memory location that is outside of the boundaries of the buffer. A subsequent read operation then produces undefined or unexpected results.



Leakage of Sensitive Info in Re-Used Buffer
Buffer contents after first HTTP request:
" p a s s w o r d " : " h u n t e r 2 "
Buffer contents after second HTTP request (from a different client):
" s o r t " : " i d " } h u n t e r 2 "
Upper bound for reading: most recently written location





Unpatched Jetty
Attacker
Jetty Webserver
Http Parser
Error Handler
Request with Malformed Header
Parse Headers
Build Error Message
Attacker Receives Sensitive Data:
Error Message with Sensitive Data

```
400 Illegal character 0x0 in state=HEADER_VALUE in 'POST /test-spec/t...ity\r\nR
eferer: \x00<<<\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00...\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\r\n\r\n>>>: password=secret..
.\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

https://insights.sei.cmu.edu/blog/inference-of-memory-bounds-preventing-the-next-heartbleed/

3. Use After Free or Dangling Pointer

Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code.

The use of previously-freed memory can have any number of adverse consequences, ranging from the corruption of valid data to the execution of arbitrary code, depending on the instantiation and timing of the flaw. The simplest way data corruption may occur involves the system's reuse of the freed memory. Use-after-free errors have two common and sometimes overlapping causes:

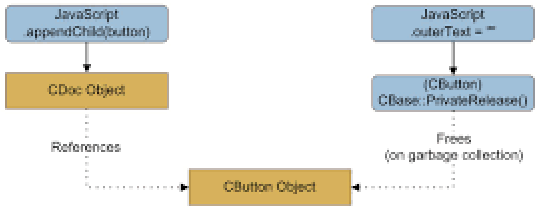
- Error conditions and other exceptional circumstances.
- Confusion over which part of the program is responsible for freeing the memory.

In this scenario, the memory in question is allocated to another pointer validly at some point after it has been freed. The original pointer to the freed memory is used again and points to somewhere within the new allocation. As the data is changed, it corrupts the validly used memory; this induces undefined behavior in the process.

If the newly allocated data happens to hold a class, in C++ for example, various function pointers may be scattered within the heap data. If one of these function pointers is overwritten with an address to valid shellcode, execution of arbitrary code can be achieved.

Scope	Impact
Integrity	**Technical Impact:** *Modify Memory* The use of previously freed memory may corrupt valid data, if the memory area in question has been allocated and used properly elsewhere.
Availability	**Technical Impact:** *DoS: Crash, Exit, or Restart* If chunk consolidation occurs after the use of previously freed data, the process may crash when invalid data is used as chunk information.

Integrity Confidentiality Availability	**Technical Impact:** *Execute Unauthorized Code or Commands* If malicious data is entered before chunk consolidation can take place, it may be possible to take advantage of a write-what-where primitive to execute arbitrary code.



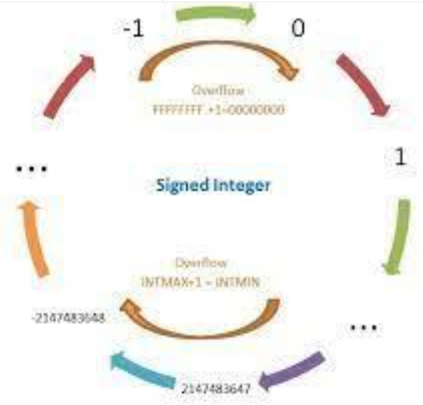
4. Integer Overflow or Wraparound

The product performs a calculation that can produce an integer overflow or wraparound, when the logic assumes that the resulting value will always be larger than the original value. This can introduce other weaknesses when the calculation is used for resource management or execution control.

An integer overflow or wraparound occurs when an integer value is incremented to a value that is too large to store in the associated representation. When this occurs, the value may wrap to become a very small or negative number. While this may be intended behavior in circumstances that rely on wrapping, it can have security consequences if the wrap is unexpected. This is especially the case if the integer overflow can be triggered using user-supplied inputs. This becomes security-critical when the result is used to control looping, make a security decision, or determine the offset or size in behaviors such as memory allocation, copying, concatenation, etc

Scope	**Impact**

Availability	**Technical Impact:** *DoS: Crash, Exit, or Restart; DoS: Resource Consumption (CPU); DoS: Resource Consumption (Memory); DoS: Instability* This weakness will generally lead to undefined behavior and therefore crashes. In the case of overflows involving loop index variables, the likelihood of infinite loops is also high.
Integrity	**Technical Impact:** *Modify Memory* If the value in question is important to data (as opposed to flow), simple data corruption has occurred. Also, if the wrap around results in other conditions such as buffer overflows, further memory corruption may occur.
Confidentiality Availability Access Control	**Technical Impact:** *Execute Unauthorized Code or Commands; Bypass Protection Mechanism* This weakness can sometimes trigger buffer overflows which can be used to execute arbitrary code. This is usually outside the scope of a program's implicit security policy.



5. Use of Hard-coded Credentials

The product contains hard-coded credentials, such as a password or cryptographic key, which it uses for its own inbound authentication, outbound communication to external components, or encryption of internal data.

Hard-coded credentials typically create a significant hole that allows an attacker to bypass the authentication that has been configured by the product administrator. This hole might be difficult for the system administrator to detect. Even if detected, it can be difficult to fix, so the administrator may be forced into disabling the product entirely. There are two main variations:

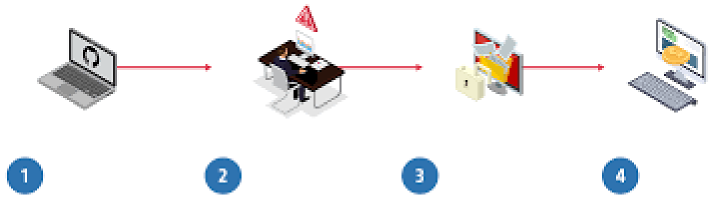
Inbound: the product contains an authentication mechanism that checks the input credentials against a hard-coded set of credentials.

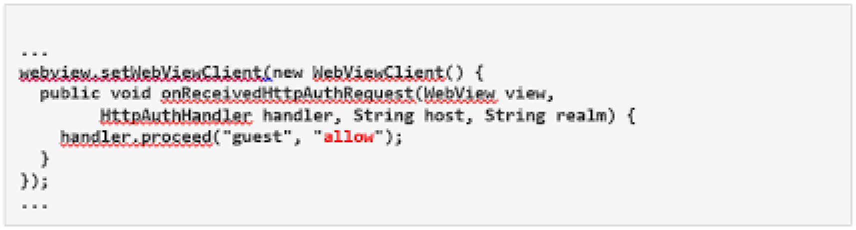
Outbound: the product connects to another system or component, and it contains hard-coded credentials for connecting to that component.

In the Inbound variant, a default administration account is created, and a simple password is hard-coded into the product and associated with that account. This hard-coded password is the same for each installation of the product, and it usually cannot be changed or disabled by system administrators without manually modifying the program, or otherwise patching the product. If the password is ever discovered or published (a common occurrence on the Internet), then anybody with knowledge of this password can access the product. Finally, since all installations of the product will have the same password, even across different organizations, this enables massive attacks such as worms to take place.

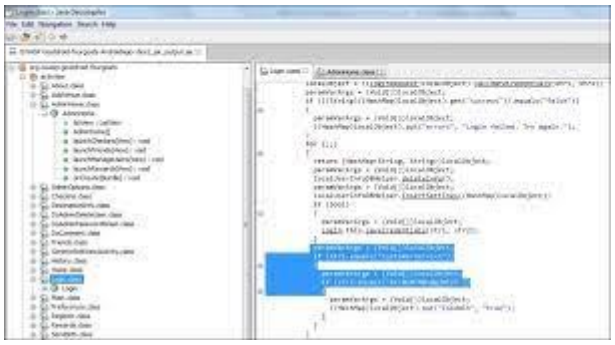
The Outbound variant applies to front-end systems that authenticate with a back-end service. The back-end service may require a fixed password which can be easily discovered. The programmer may simply hard-code those back-end credentials into the front-end product. Any user of that program may be able to extract the password. Client-side systems with hard-coded passwords pose even more of a threat, since the extraction of a password from a binary is usually very simple.

Scope	Impact	Likelihood
Access Control	**Technical Impact:** *Bypass Protection Mechanism* If hard-coded passwords are used, it is almost certain that malicious users will gain access to the account in question.	
Integrity Confidentiality Availability Access Control Other	**Technical Impact:** *Read Application Data; Gain Privileges or Assume Identity; Execute Unauthorized Code or Commands; Other* This weakness can lead to the exposure of resources or functionality to unintended actors, possibly providing attackers with sensitive information or even execute arbitrary code.	





```
...
webview.setWebViewClient(new WebViewClient() {
  public void onReceivedHttpAuthRequest(WebView view,
        HttpAuthHandler handler, String host, String realm) {
    handler.proceed("guest", "allow");
  }
});
...
```

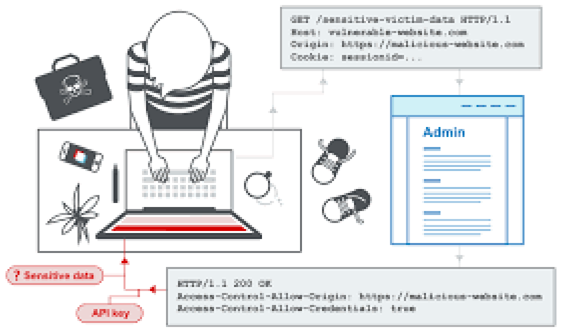


6. Cross-origin Resource Sharing (CORS) Policy

Every web-based application uses a URL as a way to connect the user's browser to its server. One common protection is called a Same Origin Policy. According to this, the server will only respond to a URL that has the same protocol, top-level domain name, and path schema. This means that you can access http://company.com/page1 and http:/company.com/page2 because they both have the following in common:

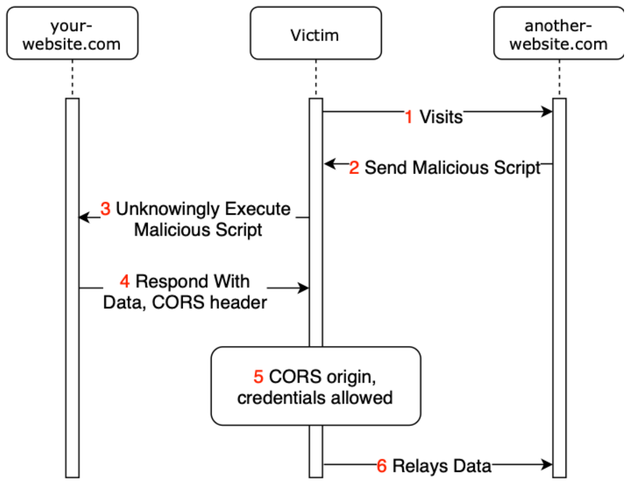
- Protocol: HTTP
- Domain: Company.com
- Path schema: /page#

Although secure, the Same Origin Policy becomes restrictive when working with web-based applications that need access to resources that connect to subdomains or third-parties.
A CORS policy gives the browser permission to access these shared resources by creating a set of allowed HTTP headers considered "trusted." For example, an application may need to pull data from two databases on different web servers. Creating a specific "allowed" list becomes too much work as you add more servers. Since the application is "shared" by both servers, the organization creates a CORS policy that lets browsers connect to both. However, if a CORS policy is not well defined, then the policy might allow the servers to provide access when a malicious actor requests it.



```
GET /sensitive-victim-data HTTP/1.1
Host: vulnerable-website.com
Origin: https://malicious-website.com
Cookie: sessionid=...
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://malicious-website.com
Access-Control-Allow-Credentials: true
```


your-website.com
Victim
another-website.com
1 Visits
2 Send Malicious Script
3 Unknowingly Execute Malicious Script
4 Respond With Data, CORS header
5 CORS origin, credentials allowed
6 Relays Data

7. HTTP verb tampering

HTTP is the protocol that lets applications respond to requests and retrieve data. An HTTP verb is one of several actions that the application can use when querying the server. Common ones HTTP verbs include:

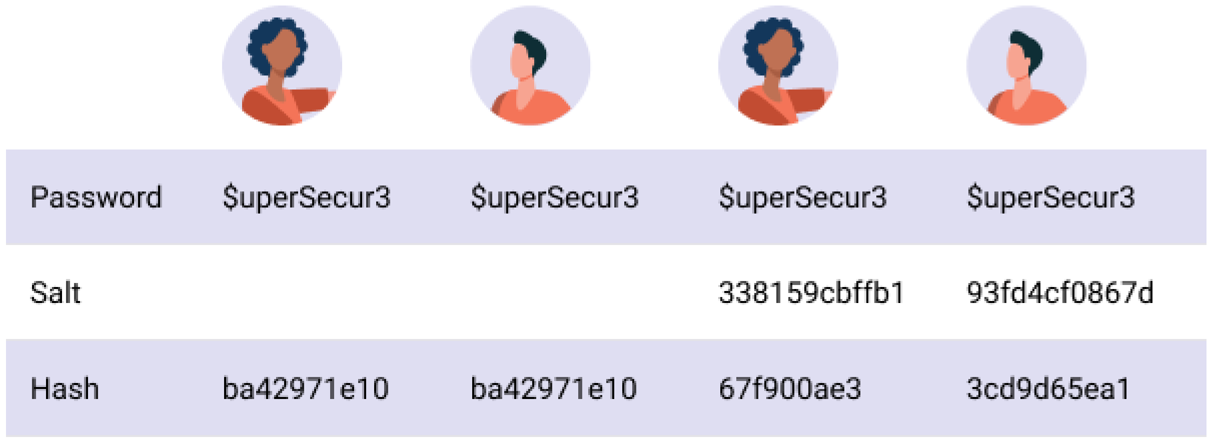
- GET: retrieves data from specified source
- HEAD: requests preview of specified resource
- POST: submits entity to specified resource, such as editing data
- PUT: transmits new data to the specified resource replacing the old information
- DELETE: deletes the specified resource entirely

Most web applications use HTTP verbs to authenticate users and manage access privileges. Malicious actors can bypass authentication and access controls intended to protect privileged information.

8. Insecure Digest

Another cryptographic vulnerability, an insecure message-digest vulnerability reduces the effectiveness of encryption. A message-digest contains the cryptographic hash function, which is the algorithm that maps an arbitrary length of data to the fixed bit array, a way of compactly storing data. Unlike encryption that requires the sender and user to have keys, hash functions do not.

Malicious actors leverage insecure digest vulnerabilities to engage in a “hash collisions attack.” The goal of the attack is to see if sending an input results in generating a duplicative hash. If the attackers brute force a shared hash, then they can offer a malicious file for download using this hash, which leaves the end-user assuming that the file is valid.



Password	$uperSecur3	$uperSecur3	$uperSecur3	$uperSecur3
Salt			338159cbffb1	93fd4cf0867d
Hash	ba42971e10	ba42971e10	67f900ae3	3cd9d65ea1

9. Missing PT_Deny_Attach

Although a bit more specific and technical than other web application vulnerabilities, this one is increasingly important as companies build out more mobile applications. A debugger is a program that helps application developers find errors in their coding. They often use debuggers to keep the application to prevent downtime from errors. However, malicious actors can leverage these same debuggers to learn how the application works and find ways to exploit them.

Process trace, more commonly called ptrace, is a system call that many debuggers and code analysis tools use. However, ptrace calls give tools a way to control their targets. The PT_DENY_ATTACH is a command for iOS mobile applications that prevents debuggers from attaching to applications. A missing PT_DENY_ATTACH command leaves an iOS mobile application at risk because malicious attackers can launch ptrace, connect to the application, and infiltrate it.

10. Race Condition

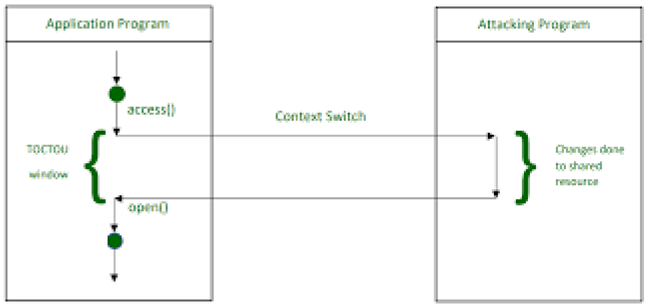
Web application processes generally rely on a series of actions, run in order, to do a task. For example, consider the following process:

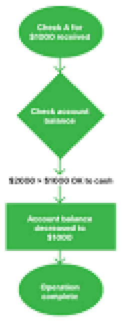
1. Click Word icon
2. Wait for Word to open
3. Click “open file”
4. Wait for list of file storage locations
5. Look for file name you want
6. Click file with the right name
7. Wait for file to open in Word
8. Edit document

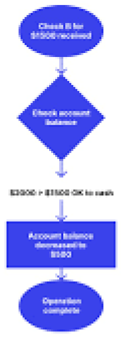
You need to do these steps in that precise order so that you can write in a new Word document. Functionally, many applications rely on a similar approach, where each step relies on the completion of the previous one.

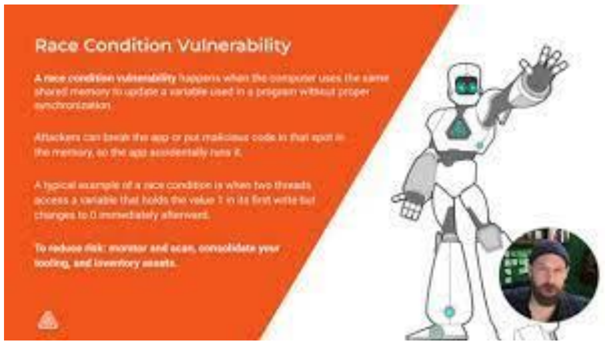
However, application tasks are often more complex and need to be faster. This means that they use multi-threaded and asynchronous order. For example, if you’re collaborating in real-time with a co-worker on a document in a shared drive, you’re both giving the application tasks. This is where the race condition vulnerability comes into play.

Incorrectly coded web applications might have logic adjusting for asynchronous actions but lack the appropriate controls. When this happens, malicious attackers can manipulate the timing of actions, which throws off the sequencing and leads to unexpected, often maliciously intended, application behaviors.









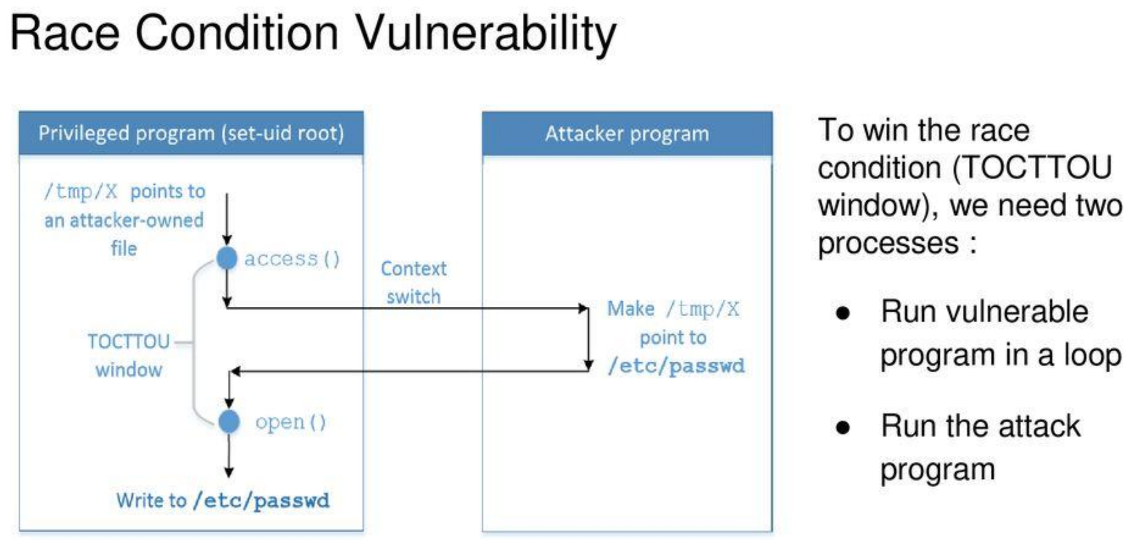
Race Condition Vulnerability

A race condition vulnerability happens when the computer uses the same shared memory to update a variable used in a program without proper synchronization.

Attackers can break the app or put malicious code in that spot in the memory, so the app accidentally runs it.

A typical example of a race condition is when two threads access a variable that holds the value 1 in its first write but changes to 0 immediately afterward.

To reduce risk: monitor and scan, consolidate your tooling, and inventory assets.



Race Condition Vulnerability

To win the race condition (TOCTTOU window), we need two processes :

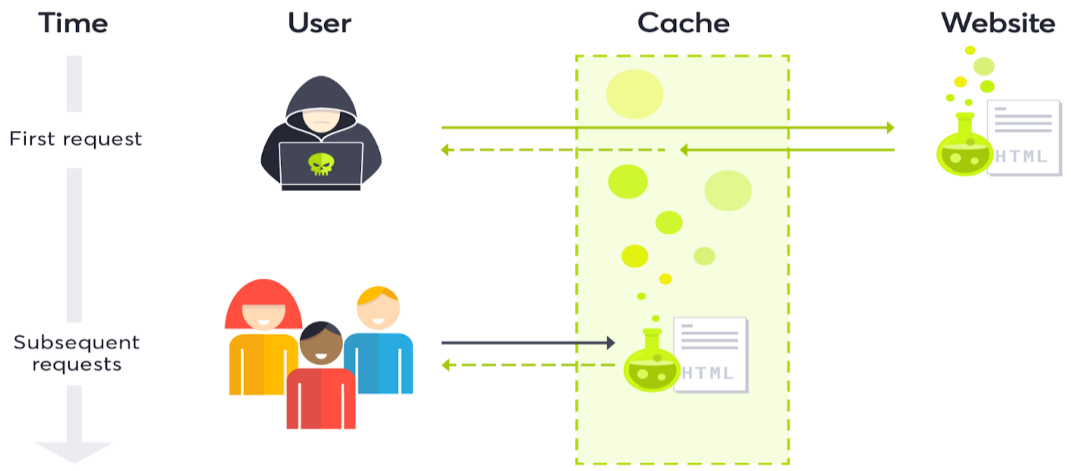
- Run vulnerable program in a loop
- Run the attack program

Privileged program (set-uid root)
Attacker program
/tmp/X points to an attacker-owned file
access()
Context switch
Make /tmp/X point to /etc/passwd
TOCTTOU window
open()
Write to /etc/passwd

Task5

Task 5
Web Server Attacks

1. **<u>Web Cache Poisoning Attack</u>**
    a. Attacks the reliability of intermediate web cache sources. The attacker swaps cached content for random URLs with infected content. Users using the online cache source can unknowingly use the poisoned content when requesting URL through online cache.
    b. Attacker forces online server's cache to flush its actual cache content and send specially crafted requests to store in cache. All users will get malicious content until servers flush online cache.
    c. It is possible if an online server and application has HTTP Response-Splitting flaws.



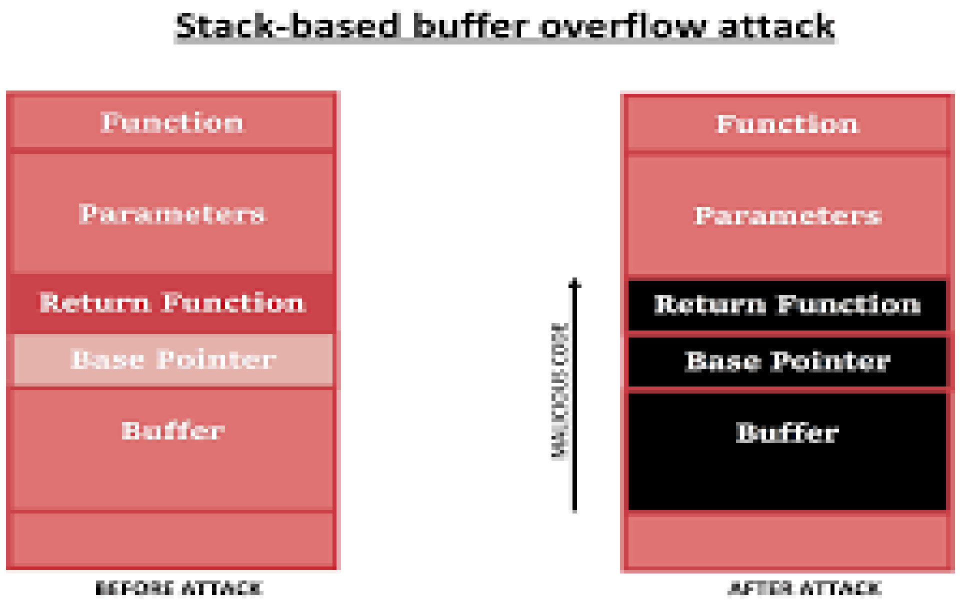
2. **<u>ASCII text file disclosure attack</u>**
a. Source code disclosure can allow attackers to realize sensitive information about database credentials and secret keys and compromise online servers.
b. Due to typographical error in scripts or due to misconfiguration, like failing to grant executable permissions to script or directory.
3. **<u>Parser Evasion Attacks</u>**
. Insecure string parsing allows attackers to remotely execute commands on machines running on a web server. If CGI (Computer Generated Imagery) script or web server features do not check for various characters in a string, an attacker can append commands to a normal value and have commands executed on vulnerable servers.

4. **<u>General Exploitation</u>**
a. Buffer overflow, format bugs, parser problems and various other attacks will contain similar data. Exploits that execute a command shell will almost always have the string "cmd.exe" in exploiting data.
b. By checking for common attacker "payloads" involved with these exploits, we can prevent an attacker from gaining unauthorized access to a web server and its data.
5. **<u>Source disclosure attack</u>**
. This type of attack can uncover passwords, web design, or business logic.

a. Example, +.htr exploit, because of vulnerabilities in the ISM.dll, IIS4 and IIS5 can be made to disclose source data, rather than executing it. An attacker can accomplish this by appending +.htr to global .asa file, netcat can help exploit this vulnerability

6. <u>Buffer Overflow Attacks</u>

. Buffer overflow vulnerabilities stem from problems in string handling. When a computer program tries copying a string or buffer into a buffer that is smaller than itself, an overflow is sometimes caused. If the buffer overflows sufficiently it will overwrite crucial system data. In most cases, an attacker can leverage this to take over a specific program's process and acquire privileges that process or program has.



7. <u>Website defacement</u>

a. It refers to unauthorized changes made to content of one website or an entire website, resulting in changes to visual appearance of the website or an internet page. Hackers forced an entry of web servers and altered hosted websites by injecting code so as to feature images, popups or text to a page in such a way that the visual appearance of the page changes. In some cases, attackers may replace the whole website rather than just changing single pages.

b. Defaced pages expose visitors to some propaganda or misleading information until unauthorized changes are discovered and corrected. Attackers use methods like MySQL injection to access an internet site to deface it.

c. To change the visual appearance of the target website, attackers deface websites for infecting the computers of visitors by making the web site vulnerable to virus attacks. Thus, website defacement is not only embarrassed by changing the appearance of its website but also harms its visitors.

8. Directory traversal attack

a. Poorly patched or configured web server software can make the online server itself vulnerable to a directory traversal attack.

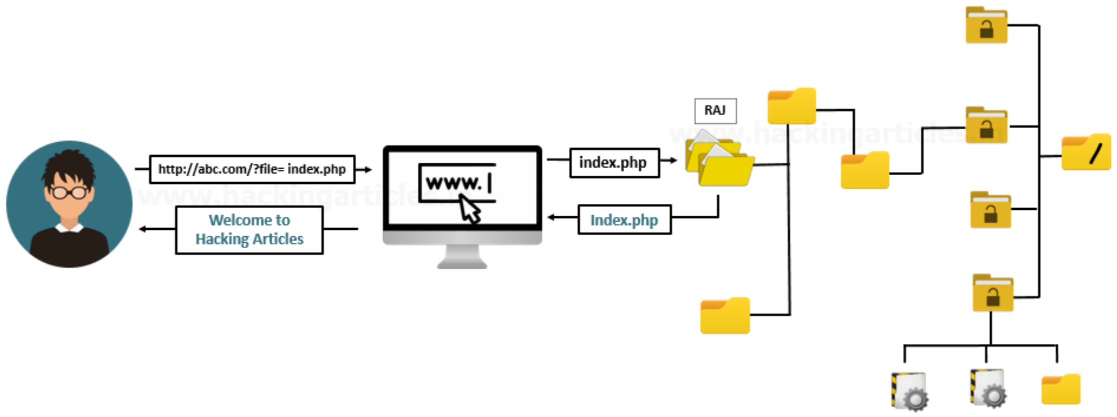
b. The design of web servers limits public access to some extent. Directory traversal is exploitation of HTTP through which attackers can access restricted directories and execute commands outside the online server's root directory by manipulating a URL.

c. Attackers use ../ (dot-dot-slash) sequence to access restricted directories outside the online server's root directory.

d. Trial-and-error method is used to navigate outside the directories and access information within the system.

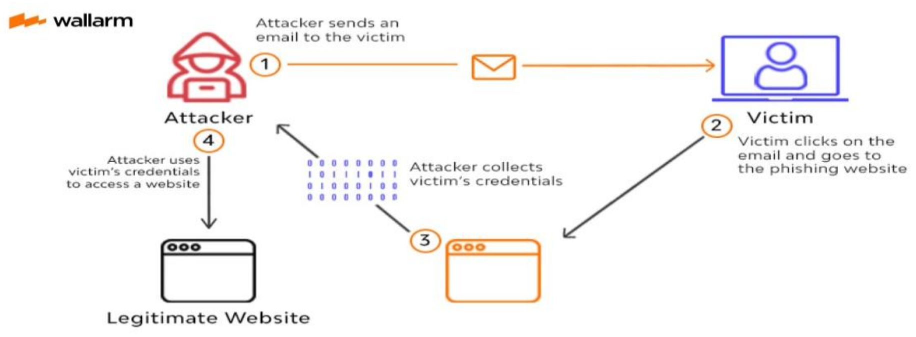
e. Attacker exploits software (web server program) on online server to perform directory traversal attacks, with assistance of browser.

f. Application fails to filter user inputs injected in legitimate commands executed by application. Using the user's input without filtering it in a legitimate command gives the attacker the possibility to execute system commands.



9. Phishing attacks

a. Attackers perform phishing attacks by sending mails containing malicious links and tricking the user to click it. The link redirects the user to a fake website that appears almost like a legitimate website. The attacker creates such websites using their address hosted on web servers. When a victim clicks on a malicious link believing the link to be legitimate, it redirects to a malicious website hosted on the attacker's server. The website prompts the user to enter sensitive information. Later the attacker could also be ready to establish a session with a legitimate website with the victim's stolen credentials.


wallarm
Attacker sends an email to the victim
1
Attacker
Victim
2
Victim clicks on the email and goes to the phishing website
4
Attacker uses victim's credentials to access a website
Attacker collects victim's credentials
3
Legitimate Website

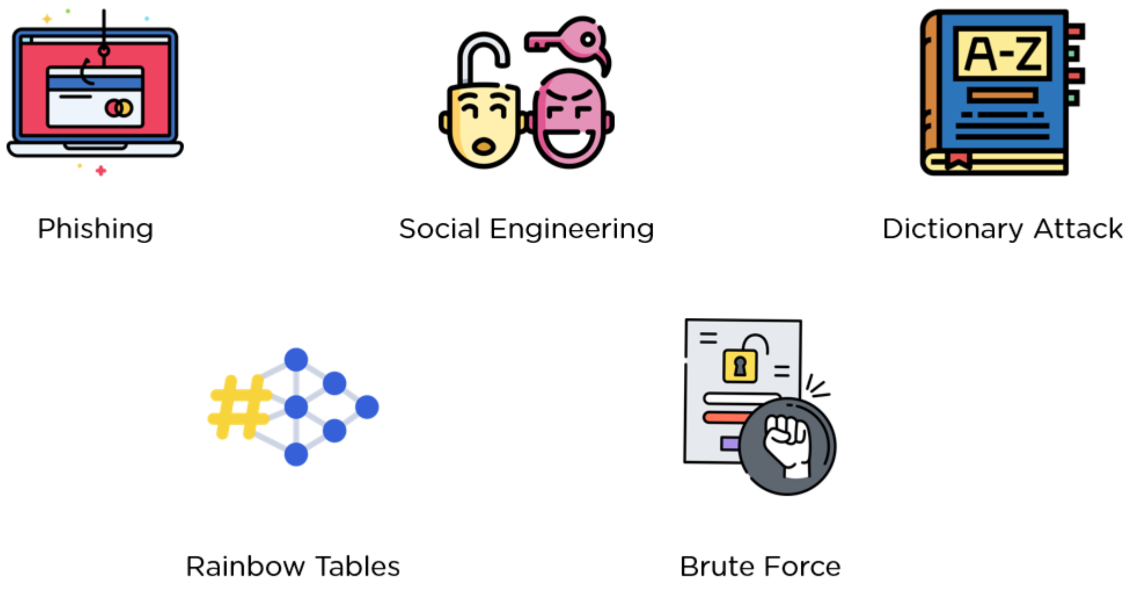
10. <u>Web Server Password cracking</u>

a. An attacker tries to exploit weaknesses to hack passwords like password, root, administrator, admin, demo, test, guest, qwerty, pet names etc….

b. SMTP and FTP servers, Web servers, SSH tunnels, web form authentication cracking

c. Attackers use different methods like social engineering, spoofing, phishing, trojan horse or virus, wiretapping, keystroke logging etc..

d. Attackers use cracked passwords to prove to the server that they're legitimate users.



Task6

1. Manage all hardware devices on network so that only required devices are given access and unauthorized and unmanaged devices are found and prevented from gaining access. (inventory,track and correct)

It’s so difficult to find the devices on network and eliminate the ones u don’t want. So we prevent the connection in the first place. This process is Network Access control (NAC).

You can’t prevent people from connecting rogue devices into open ports on the wall, but can restrict them from sending information through your network.

Use tool to actively discover and monitor devices connected to network and make sure hardware assets are up to date and new security, makes it easier to identify threats. Authenticate devices to prevent unnecessary access.

2. Actively manage all software like OS and applications on the network

Only authorized software is found and unauthorized are or unmanaged software is prevented from installation or execution

Attackers seek for vulnerable versions of software that can be remotely exploited, like they seek for any way(access to malicious website or attachment with vulnerable browser) to install backdoor programs and bots to gain long-term control over the system. So complete inventory of software assets is essential.

3. Continuous vulnerability management

Develop a plan (automated) to continuously assess and track vulnerabilities on all the enterprise assets.

Monitor public and private industry sources for new threat and vulnerability to prevent any attack.

Continuous monitoring reduces burden on security teams and reduces mean time to resolution.

4. **Controlled Use of Administrative Privileges**

All user with administrator level access use dedicated or secondary account which is not accessible for other purpose like web browsing, email or by other users. This is done to prevent users from modifying, customizing or erasing sensitive information.

Misuse of this account can cause attacker to spread inside the network. Passwords set for this should be strong and should not be used in any other critical systems.

5. **Secure Configuration for Hardware and Software on Mobile**

Maintain documented security configuration standards for all OS and s/w.

Maintain a security configuration, implement configuration management and change control process, and actively report on security configuration of all endpoint devices mobile devices, laptops, servers and workstations.

Default configuration of end devices is set up for ease-of use and not towards security.

Basic controls, open services and ports, default account and passwords, older protocols and pre-installation of unwanted s/w are vulnerable.

Strong initial configuration and continuous management is essential.

6. Maintenance, Monitoring and Analysis of Audit Logs

Collect, manage and analyze audit logs of events that helps to detect and prevent attack, understand, or recover from an attack.

Any deficiencies in log allow attackers to hide their location, malicious software and activities. Without log details it's difficult to find details of attack and any further actions of attacker, even if it is known that the system is compromised. Sometimes an attack can go unnoticed and in such cases damages can be irreversible. Logging records can be the sole evidence of a successful attack, it is imp to look up the records frequently.

7. Email and Web Browser Protections

Protect against or detect the threats from email and web vectors

This gives chance for attackers to directly engage with people from within the enterprise. (social engineering attacks).

So it has to be protected, some methods are:

Use of fully supported browsers and email clients (latest version) throughout organization, DNS filtering services, restricting unnecessary or unauthorized browser and email client extensions, maintain and enforce network-based URL filters and so on…

8. **Malware Defenses**

Control installation, spread and execution of malicious code

Using automation to enable rapid updating of defense, data gathering and corrective action.

Malware is generally fast-moving, fast-changing and enters via any no of points like end-user device, attachment, web page, cloud services, user action and removable media. They are designed to avoid defenses and attack or disable them.

Thus it is important to operate them in dynamic environment through large-scale automation, rapid updating and integration with processes like incident response.

9. Limitation and Control of Network Ports, Protocols, and Services

Manage the operational use of ports, protocols, and services on devices on the network.

Attackers search for remotely accessible network services that are vulnerable, so it is imp to ensure only validated business required ports, protocols and services are running.

Perform automated scans on regular basis to ensure that unauthorized ports/services are detected if open.

10 . Data Recovery Capabilities

Ensure that processes and tools used to properly backup information

Methodology for timely recovery of data is also important as backing them up

When there is some attack, significant changes are made to configurations and software and sometimes subtle changes on data stored.

To recover from this and remove all aspects of attackers presence on the machine it is imp to have a trustworthy data recovery.

11. Secure Configuration for Network Devices, such as Firewalls, Routers and Switches

Establish, implement, actively manage security configurations of network infrastructure devices using rigorous configuration management and change control process.

Default configuration-easy to us and not secure. Secure configuration should be performed regularly re-evaluating configuration items and traffic flow.

12. **Boundary Defense**

Ensure that entry points into network are clearly defined and monitored. Network boundaries do not have a clear edge and no longer defined as single entry point protected by firewall and edge routers.

The network perimeter is beyond gateway like the cloud when using AWS or ASURE, wireless network, VPN endpoints with more remote jobs (extranet). Understanding each network and risks associated with it is essential.

13. **Data Protection**

Develop processes and technical controls to identify, classify, securely handle, retain, and dispose data.

Data is now on cloud, on portable end-user device, shared with partners or online services. Data privacy has become important. Data must be appropriately used and managed through its life-cycle following privacy rules.

Monitoring data outflow is important as attackers try to exfiltrate data.

Adoption of data encryption can provide mitigate(make it less severe) against data compromise.

14. Controlled Access Based on the Need to Know

Control such that users are allowed to access only information they are authorized or need to perform job duties. This is implemented using layers like network segmentation, data classification and use of Data Loss Prevention (DLP) products.

This tries to minimize exposure of data to attackers and proper management of data.

15. **Wireless Access Control**

Ensure wireless access is configured to track and control access, prevent unauthorized access.

Manage the configuration settings.

Major attacks are by attackers who have gained wireless access to organizations from outside it. Wireless clients accompanying travelers are infected on a regular basis through remote exploitation while on public wireless networks found in airports and cafes. These infected ones are used as backdoors when they are reconnected to the network of a target organization.

Because they do not require direct physical connections, wireless devices are a convenient vector for attackers to maintain long-term access into a target environment.

16. **Account Monitoring and Control**

Ensure that all accounts are managed in a fashion that promotes clean account. This misuse or neglect of account maintenance can lead to system compromise.

Attackers exploit legitimate but inactive user (terminated employees or contractors, accounts formerly set up for Red Team testing) accounts impersonating legitimate user, this makes it difficult for security personnel watchers to identify the attacker behavior and makes it easier for attackers.

17. **Implement a Security Awareness and Training Program**

Establish and maintain a security awareness program to influence behavior among the workforce to be security conscious and properly skilled to reduce cybersecurity risks to the enterprise.

For all functional roles in organization identify specific knowledge, skills and abilities needed to support defense of the enterprise.

Develop and execute integrated plan to assess, identify gaps and remediate through policy, organizational planning, training, and awareness programs.

18. **Application Software Security**

Manage the security life cycle of in-house developed, hosted, or acquired software to prevent, detect, and remediate security weaknesses before they can impact the enterprise.

Applications are used by both user (access and manage data, easy log in to database) and enterprises (to manage sensitive data and control access).

Therefore, an attacker can use the application itself to compromise the data, instead of an elaborate network and system hacking sequence that attempts to bypass network security controls and sensors.

Lacking credentials and application flaws are attack vectors.

19. Incident Response and Management

a. Establish a program to develop and maintain an incident response capability (e.g., policies, plans, procedures, defined roles, training, and communications) to prepare, detect, and quickly respond to an attack to prevent spread and reduce any harm.
b. A comprehensive cybersecurity program includes protections, detections, response and recovery capabilities.
c. If an incident occurs despite the protections, an effective documented plan is important along with good people to investigate, report and collect data, manage legal protocols and responsibilities, communication strategy for successful understanding, management and recovery.

d. Reduce the impact of attack, find any traces of the attacker and eradicate them, prevent any further harm, and sanitize affected devices.

20. **Penetration Tests and Red Team Exercises**

Test all security controls, train for security awareness.

A really strict password policy can result in users taping passwords to their keyboard. A great technical control, thwarted by a forgetful user and an observant adversary. Many times developers find protocols they find useful, and never realize there is an inherent security flaw, for example FTP and Telnet, are great tools. But in both cases, all credential exchanges are in clear text, allowing passwords and other information to be captured easily. Many chat programs use a form of HTTP and not HTTPS, again data is exchanged in the clear. With wireless technologies, many times with a simple wireless receiver, anyone can monitor the full exchanges of information. Penetration tests and red team exercises help to bring this information to the forefront of the security conversation.