Secure Programming and Scripting

Assignment 1:

Killian Halpin

20093283.

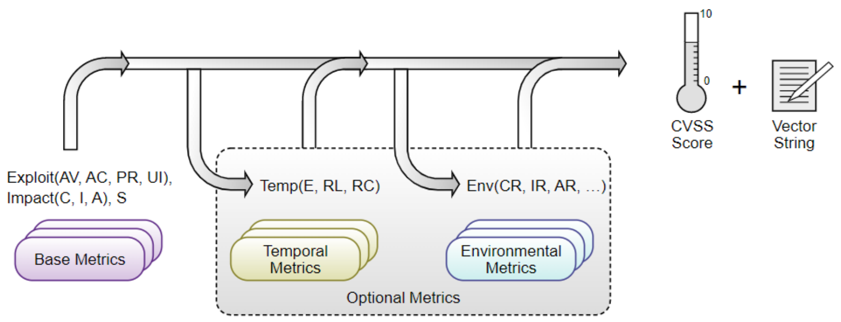
Criteria for selecting the most important breaches:

The common vulnerability scoring system is an open framework for deciding the characteristics and severity for software vulnerabilities it consists of three metric groups **Base, Temporal** and **Environmental.** The CVSS is owned by First.Org.inc a non-profit organisation.

The base score of the CVSS reflects the severity of the vulnerability which are constant overtime, Temporal Metrics adjusts the Base severity of the vulnerability on factors that change overtime. Environmental metrics adjusts both base and temporal severities to specific computing environment.

Scoring:

When the base metrics are assigned values the equation which is shown below gives it a score between 0 and 10. This is how we decide on the severity of the attack.



The CVSS scoring also creates a vector string a text value of the matric values used to score the vulnerabilities. All matrices should he scored with the assumption that the attacker has already found the vulnerability.

There could be many people doing this matric scoring, but the scoring is intended to be agnostic to the individual and their organisation.

For a breach to be considered most important there are many metrics it must go through to test whether the threat is of importance or not. The diagrams below show how the threat is analysed, and they are used in calculating the risk for the CVSS.

Metric Value	Description
Network (N)	The vulnerable component is bound to the network stack and the set of possible attackers extends beyond the other options listed below, up to and including the entire Internet. Such a vulnerability is often termed "remotely exploitable" and can be thought of as an attack being exploitable *at the protocol level* one or more network hops away (e.g., across one or more routers). An example of a network attack is an attacker causing a denial of service (DoS) by sending a specially crafted TCP packet across a wide area network (e.g., CVE-2004-0230).
Adjacent (A)	The vulnerable component is bound to the network stack, but the attack is limited *at the protocol level* to a logically adjacent topology. This can mean an attack must be launched from the same shared physical (e.g., Bluetooth or ILLL 802.11) or logical (e.g., local IP subnet) network, or from within a secure or otherwise limited administrative domain (e.g., MPLS, secure VPN to an administrative network zone). One example of an Adjacent attack would be an ARP (IPv4) or neighbor discovery (IPv6) flood leading to a denial of service on the local LAN segment (e.g., CVE-2013-6014).
Local (L)	The vulnerable component is not bound to the network stack and the attacker's path is via read/write/execute capabilities. Either: ▪ the attacker exploits the vulnerability by accessing the target system locally (e.g., keyboard, console), or remotely (e.g., SSH); *or* ▪ the attacker relies on User Interaction by another person to perform actions required to exploit the vulnerability (e.g., using social engineering techniques to trick a legitimate user into opening a malicious document).
Physical (P)	The attack requires the attacker to physically touch or manipulate the vulnerable component. Physical interaction may be brief (e.g., evil maid attack[^1]) or persistent. An example of such an attack is a cold boot attack in which an attacker gains access to disk encryption keys after physically accessing the target system. Other examples include peripheral attacks via FireWire/USB Direct Memory Access (DMA).

Table 1: Attack Vector

Metric Value	Description
Low (L)	Specialized access conditions or extenuating circumstances do not exist. An attacker can expect repeatable success when attacking the vulnerable component.
High (H)	A successful attack depends on conditions beyond the attacker's control. That is, a successful attack cannot be accomplished at will, but requires the attacker to invest in some measurable amount of effort in preparation or execution against the vulnerable component before a successful attack can be expected.[^2] For example, a successful attack may depend on an attacker overcoming any of the following conditions: ▪ The attacker must gather knowledge about the environment in which the vulnerable target/component exists. For example, a requirement to collect details on target configuration settings, sequence numbers, or shared secrets. ▪ The attacker must prepare the target environment to improve exploit reliability. For example, repeated exploitation to win a race condition, or overcoming advanced exploit mitigation techniques. ▪ The attacker must inject themselves into the logical network path between the target and the resource requested by the victim in order to read and/or modify network communications (e.g., a man in the middle attack).

Table 2: Attack Complexity

Metric Value	Description
None (N)	The attacker is unauthorized prior to attack, and therefore does not require any access to settings or files of the vulnerable system to carry out an attack.
Low (L)	The attacker requires privileges that provide basic user capabilities that could normally affect only settings and files owned by a user. Alternatively, an attacker with Low privileges has the ability to access only non-sensitive resources.
High (H)	The attacker requires privileges that provide significant (e.g., administrative) control over the vulnerable component allowing access to component-wide settings and files.

Table 3: Privileges Required

Metric Value	Description
None (N)	The vulnerable system can be exploited without interaction from any user.
Required (R)	Successful exploitation of this vulnerability requires a user to take some action before the vulnerability can be exploited. For example, a successful exploit may only be possible during the installation of an application by a system administrator.

Table 4: User Interaction

Metric Value	Description
Unchanged (U)	An exploited vulnerability can only affect resources managed by the same security authority. In this case, the vulnerable component and the impacted component are either the same, or both are managed by the same security authority.
Changed (C)	An exploited vulnerability can affect resources beyond the security scope managed by the security authority of the vulnerable component. In this case, the vulnerable component and the impacted component are different and managed by different security authorities.

Table 5: Scope

Metric Value	Description
High (H)	There is a total loss of confidentiality, resulting in all resources within the impacted component being divulged to the attacker. Alternatively, access to only some restricted information is obtained, but the disclosed information presents a direct, serious impact. For example, an attacker steals the administrator's password, or private encryption keys of a web server.
Low (L)	There is some loss of confidentiality. Access to some restricted information is obtained, but the attacker does not have control over what information is obtained, or the amount or kind of loss is limited. The information disclosure does not cause a direct, serious loss to the impacted component.
None (N)	There is no loss of confidentiality within the impacted component.

Table 6: Confidentiality

Metric Value	Description
High (H)	There is a total loss of integrity, or a complete loss of protection. For example, the attacker is able to modify any/all files protected by the impacted component. Alternatively, only some files can be modified, but malicious modification would present a direct, serious consequence to the impacted component.
Low (L)	Modification of data is possible, but the attacker does not have control over the consequence of a modification, or the amount of modification is limited. The data modification does not have a direct, serious impact on the impacted component.
None (N)	There is no loss of integrity within the impacted component.

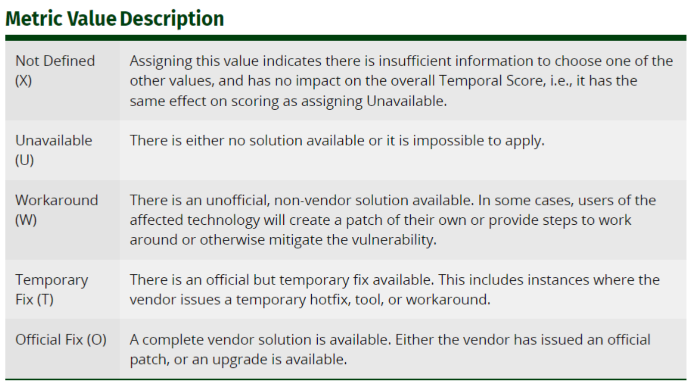
Table 7: Integrity

Metric Value	Description
High (H)	There is a total loss of availability, resulting in the attacker being able to fully deny access to resources in the impacted component; this loss is either sustained (while the attacker continues to deliver the attack) or persistent (the condition persists even after the attack has completed). Alternatively, the attacker has the ability to deny some availability, but the loss of availability presents a direct, serious consequence to the impacted component (e.g., the attacker cannot disrupt existing connections, but can prevent new connections; the attacker can repeatedly exploit a vulnerability that, in each instance of a successful attack, leaks a only small amount of memory, but after repeated exploitation causes a service to become completely unavailable).
Low (L)	Performance is reduced or there are interruptions in resource availability. Even if repeated exploitation of the vulnerability is possible, the attacker does not have the ability to completely deny service to legitimate users. The resources in the impacted component are either partially available all of the time, or fully available only some of the time, but overall there is no direct, serious consequence to the impacted component.
None (N)	There is no impact to availability within the impacted component.

Table 8: Availability

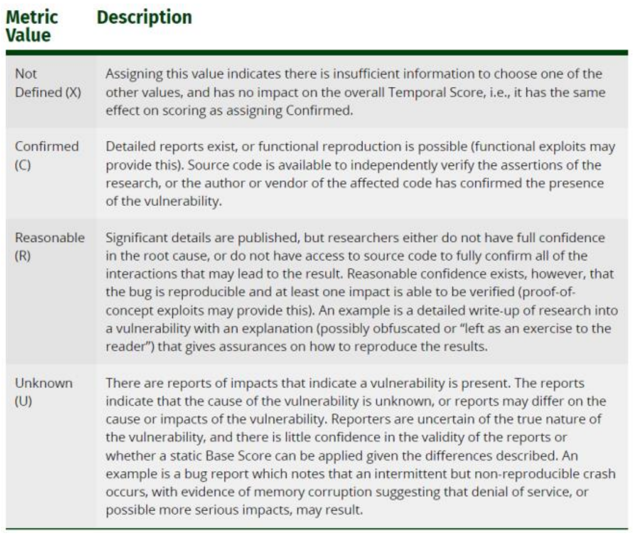
Metric Value	Description
Not Defined (X)	Assigning this value indicates there is insufficient information to choose one of the other values, and has no impact on the overall Temporal Score, i.e., it has the same effect on scoring as assigning High.
High (H)	Functional autonomous code exists, or no exploit is required (manual trigger) and details are widely available. Exploit code works in every situation, or is actively being delivered via an autonomous agent (such as a worm or virus). Network-connected systems are likely to encounter scanning or exploitation attempts. Exploit development has reached the level of reliable, widely available, easy-to-use automated tools.
Functional (F)	Functional exploit code is available. The code works in most situations where the vulnerability exists.
Proof-of-Concept (P)	Proof-of-concept exploit code is available, or an attack demonstration is not practical for most systems. The code or technique is not functional in all situations and may require substantial modification by a skilled attacker.
Unproven (U)	No exploit code is available, or an exploit is theoretical.

Table 9: Exploit Code Maturity



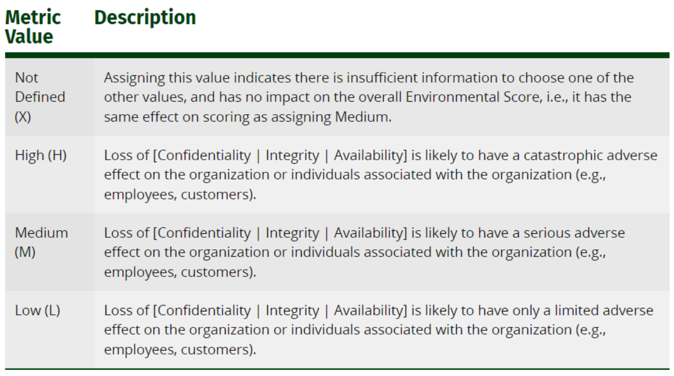
Metric Value	Description
Not Defined (X)	Assigning this value indicates there is insufficient information to choose one of the other values, and has no impact on the overall Temporal Score, i.e., it has the same effect on scoring as assigning Unavailable.
Unavailable (U)	There is either no solution available or it is impossible to apply.
Workaround (W)	There is an unofficial, non-vendor solution available. In some cases, users of the affected technology will create a patch of their own or provide steps to work around or otherwise mitigate the vulnerability.
Temporary Fix (T)	There is an official but temporary fix available. This includes instances where the vendor issues a temporary hotfix, tool, or workaround.
Official Fix (O)	A complete vendor solution is available. Either the vendor has issued an official patch, or an upgrade is available.

Table 10: Remediation Level



Metric Value	Description
Not Defined (X)	Assigning this value indicates there is insufficient information to choose one of the other values, and has no impact on the overall Temporal Score, i.e., it has the same effect on scoring as assigning Confirmed.
Confirmed (C)	Detailed reports exist, or functional reproduction is possible (functional exploits may provide this). Source code is available to independently verify the assertions of the research, or the author or vendor of the affected code has confirmed the presence of the vulnerability.
Reasonable (R)	Significant details are published, but researchers either do not have full confidence in the root cause, or do not have access to source code to fully confirm all of the interactions that may lead to the result. Reasonable confidence exists, however, that the bug is reproducible and at least one impact is able to be verified (proof-of-concept exploits may provide this). An example is a detailed write-up of research into a vulnerability with an explanation (possibly obfuscated or "left as an exercise to the reader") that gives assurances on how to reproduce the results.
Unknown (U)	There are reports of impacts that indicate a vulnerability is present. The reports indicate that the cause of the vulnerability is unknown, or reports may differ on the cause or impacts of the vulnerability. Reporters are uncertain of the true nature of the vulnerability, and there is little confidence in the validity of the reports or whether a static Base Score can be applied given the differences described. An example is a bug report which notes that an intermittent but non-reproducible crash occurs, with evidence of memory corruption suggesting that denial of service, or possible more serious impacts, may result.

Table 11: Report Confidence



Metric Value	Description
Not Defined (X)	Assigning this value indicates there is insufficient information to choose one of the other values, and has no impact on the overall Environmental Score, i.e., it has the same effect on scoring as assigning Medium.
High (H)	Loss of [Confidentiality | Integrity | Availability] is likely to have a catastrophic adverse effect on the organization or individuals associated with the organization (e.g., employees, customers).
Medium (M)	Loss of [Confidentiality | Integrity | Availability] is likely to have a serious adverse effect on the organization or individuals associated with the organization (e.g., employees, customers).
Low (L)	Loss of [Confidentiality | Integrity | Availability] is likely to have only a limited adverse effect on the organization or individuals associated with the organization (e.g., employees, customers).

Table 12: Security Requirements

These tables are paramount in the investigating of threats. The results obtained from these tables is what gives the CVSS score once you put the results into the calculator.

Breaches:

Nintendo NNID Hack:

On the 24th of April 2020 Nintendo suffered an attack. They got in using the NNID which was vulnerable as it didn't have 2 factor Authentication. The password size was only 6 characters long which made the attack even more possible as it led to much weaker passwords as the standard is 8 characters long. The attack affected 300,000 accounts. 1% of the accounts the accounts hacked were used for fraudulent trade. The data that was compromised included nickname, date of birth, country region and e-mail were revealed. Some Credit card information was revealed. The attacker gained access to the card type and currency, expiration date first 6 digits and also the last 4 digits of the card were leaked. The attacker could also see the users Nintendo gold points. The attacker could also make purchases on the Nintendo store. The attack was carried out using credential stuffing and also a custom-made program. The attacker tried using usernames and passwords acquired from other data breaches as users tend to use the same password and username for multiple sites. Spycloud gained access the source code used for this attack. It was a custom-made account checker made specifically for Nintendo accounts. The program had a defence feature. It would only let the attacker use it and no one else. The code would not work without payment, and it would only work on one computer. There was also a built-in kill switch if it detected debugging tools being used. The source code used proxies to hide the IP of the hacker. In order for the code to work the hacker had to provide it with passwords which were acquired from other data breaches. These data breaches could have happened years ago and there was a possibility that they still might have worked.

The Nintendo NNID breach could have been easily avoided if two factor authentication had been in place. The attacker would not have been able to access account information as they would not have been able to emulate the users phone. It could also have been avoided if Nintendo applied a stricter password policy making their passwords 8 characters instead of 6. This would have led to much stronger passwords and less chance the attacker could have gained access to their accounts.

I decided to write about this attack as I have a Nintendo switch and it could have very easily affected me, thankfully it didn't. There was also a lot of data breached for the 300,000 people it did affect. The attacker had enough information to go and try gain access to the users PayPal accounts. They also had enough information to try guess the last 6 digits they needed to have access to the users credit card.

Attack Details:

Attack Vector – Network

Attack Complexity – Low – had access to wide range of passwords and no 2-factor authentication.

Privileges Required – Low – no 2-factor authentication only needed unique ID and password.

User interaction – None.

Scope – Unchanged.

Confidentiality – Low.

Integrity – High – PayPal id email address some credit card information and date of birth were leaked.

Availability – None – no denial of service.

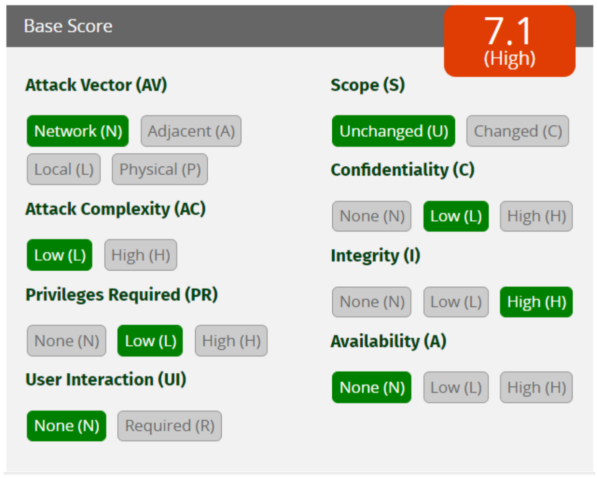
Exploit code – High – made especially for Nintendo and also had security features.

Remediation Level – High – there is an official fix in place.

Report Confidence – Confirmed.

Security Requirements – Medium.

In the end the threat generated a 7.1 base score and a temporal score of 6.8. In the end the threat only turned out to be a medium, but I still think I'm justified in picking this example as it could have a knock-on effect as peoples PayPal ID was linked and also peoples' private emails, the attacker could gain access to sensitive material or even peoples' money. This hack also hits close to home as I could have been one of the 300,000 affected but thankfully, I wasn't.



Base Score 7.1 (High)

Attack Vector (AV)
[✓ Network (N)] Adjacent (A) Local (L) Physical (P)

Attack Complexity (AC)
[✓ Low (L)] High (H)

Privileges Required (PR)
None (N) [✓ Low (L)] High (H)

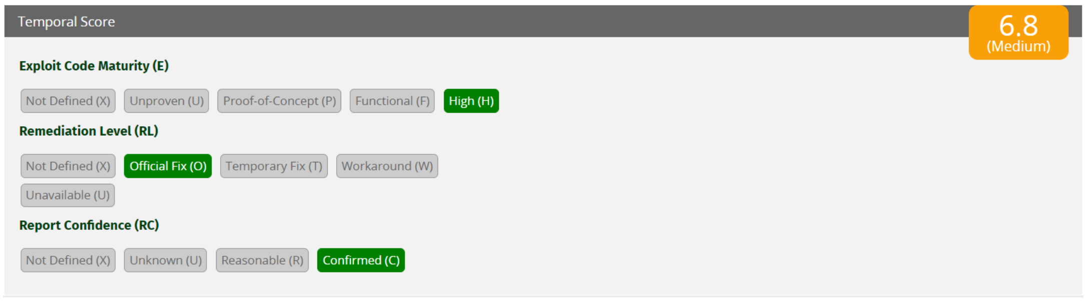
User Interaction (UI)
[✓ None (N)] Required (R)

Scope (S)
[✓ Unchanged (U)] Changed (C)

Confidentiality (C)
None (N) [✓ Low (L)] High (H)

Integrity (I)
None (N) Low (L) [✓ High (H)]

Availability (A)
[✓ None (N)] Low (L) High (H)



Temporal Score 6.8 (Medium)

Exploit Code Maturity (E)
Not Defined (X) Unproven (U) Proof-of-Concept (P) Functional (F) [✓ High (H)]

Remediation Level (RL)
Not Defined (X) [✓ Official Fix (O)] Temporary Fix (T) Workaround (W) Unavailable (U)

Report Confidence (RC)
Not Defined (X) Unknown (U) Reasonable (R) [✓ Confirmed (C)]

SolarWinds Orion Attack.

The other attack I chose is Solar Winds Orion attack. The attackers managed to infect the software updates custom-built malware which created a backdoor access into aby system which downloaded the updates. 18,000 high valued targets including every US Government agency, CISCO system and Microsoft system along with other well known organisations. The attack was executed by Russias Foreign Intelligence Service. The attack served as a warning for the rest of the world.

References:

Cobalt. 2022. *Understanding the CVSS Base Score: An Essential Guide | Cobalt Blog*. [online] Available at: <https://cobalt.io/blog/understanding-the-cvss-base-score-an-essential-guide> [Accessed 17 February 2022].

Cydrill Software Security. 2022. *The Nintendo Cyber Security Data Breach*. [online] Available at: <https://cydrill.com/cyber-security/the-nintendo-cyber-security-data-breach/> [Accessed 18 February 2022].

Cyrex. 2022. *Nintendo's NNID Hack: The Impact of Authentication Flaws - Cyrex*. [online] Available at: <https://cyrextech.net/nintendos-nnid-hack-the-impact-of-authentication-flaws/> [Accessed 18 February 2022].

FIRST — Forum of Incident Response and Security Teams. 2022. *Common Vulnerability Scoring System Version 3.1 Calculator*. [online] Available at: <https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:H/A:N/E:H/RL:O/RC:C> [Accessed 15 February 2022].

freeCodeCamp.org. 2022. *Biggest Data Breaches of 2020 – and What Developers Should Learn From Them*. [online] Available at: <https://www.freecodecamp.org/news/biggest-data-breaches-lessons-learned/> [Accessed 17 February 2022].