Activity 8: Part 2

Ryan Coon

CST-407

Professor Ty Howard

February 11, 2024

Operating Systems Security

a.Explain the terminology and how it relates to a buffer overflow exploit: (a) stack, (b) return address, (c) allocated buffer space, (d) segment fault, (e) GDB, and (f) no-op sled.

In the context of a buffer overflow exploit, the "stack" refers to a region of memory used for the storage of local variables, function parameters, and return addresses within a program's execution environment. The stack typically grows and shrinks dynamically as functions are called and return. The return address is a memory address that indicates the location to which the control flow should return after a function call or subroutine execution. When a function is called, the return address is typically stored on the call stack, which is a region of memory used for managing function calls and local variables. After the function completes its execution, the program uses the return address to transfer control back to the original calling code. An allocated memory buffer is a reserved portion of memory used for storing data within a program. The size of the allocated buffer space is determined by the program at runtime or during initialization, and it is typically based on the requirements of the specific data being stored. A segment fault is simply stated as a fault that is created by exceeding the allocated buffer size. GDB is a debugger that allows developers and security professionals to analyze, debug, and troubleshoot programs at the source code level. GDB is particularly valuable for diagnosing issues, understanding program behavior, and identifying vulnerabilities, including those related to buffer overflows and other memory-related exploits. A no-op sled, or slide, is a sequence of no-operation instructions placed in the exploited buffer, followed by the malicious code. This is helpful because memory locations change at any time, so the no-op sled helps by removing the requirement from the user to know the exact location of malicious code.

b.Code Red and SQL Slammer were events caused by a buffer overflow exploit. What was the result and cost of the attacks?

Code Red was a computer worm that used a buffer overflow exploit to spam the letter “N” in order to overflow a buffer. This attack targeted the vulnerability within software that was distributed with ISS. This was a large scale attack that cost an estimated $2.4 billion in damages. Months later Code Red II emerged, this was simply the same exploit but spammed the string of “X” instead of “N”.

The SQL Slammer was another buffer overflow worm that would select ip addresses at random and send itself out to them. If a victim that was targeted by this worm was running an unpatched copy of Microsoft SQL Server, the host immediately becomes infected and begins spraying the internet with copies of the worm. This in turn caused a Denial Of Service (DOS) hitting banks in US and Canada costing an estimated $1.2 billion in damages.

c.What languages are most often used for writing buffer exploits, and what do they have in common?

C and C++ are commonly used for developing buffer exploits due to their low-level system access and direct memory manipulation capabilities. They are frequently employed in crafting exploits that target vulnerabilities related to buffer overflows, stack smashing, and other memory-related issues. Assembly language is another common choice for creating buffer exploits, as it provides granular control over system resources and memory management. Exploit developers often use assembly language to craft precise, low-level code that can take advantage of memory vulnerabilities.

d.What are some operating system improvements that attempt to prevent intentional buffer overflow exploits?

Some operating system improvements include Address space layout randomization (ASLR), Data Execution Prevention (DEP), Stack Canaries, and Control Flow Integrity (CFI). Another I have read about Bounds checking to prevent overflow in Windows and most programming languages.

e.Based on the associate resources, why was Windows XP used as a target machine?

Windows XP was the target due to the Universal Plug and Play vulnerability. Through this vulnerability, it allowed the attacker complete control over a network of other vulnerable devices from a single session.

f.What is the Beast program and what are its capabilities?

The Beast attack is a cryptographic exploit that targets the SSL/TLS protocols, which are commonly used to secure communications over the internet, particularly for web browsing and other online transactions. This attack allows a MITM attacker to uncover information from an encrypted SSL/TLS session by exploiting a known vulnerability.  Modern versions of SSL/TLS protocols have introduced security enhancements and mitigations to address these weaknesses. Web servers and clients have also implemented patches and updates to mitigate the risk of such attacks.

g.Explain how some of the command options in FU make this tool useful as a rootkit.

This rootkit contains commands that allow many processes which make it a useful rootkit. This rootkit can hide processes and files, especially those that are malicious to keep them hidden from detection. It allows attackers to hide network connections to help the attacker maintain their stealth. It can adjust the privilege on a process and even impersonate another login session so that windows does not know who performed said actions.

h.How is the Black Light security tool shown in the video different than the normal Process List utility of Windows?

The Black Light security tool can scan for processes hidden to Windows Task Manager. The video shows the attacker launching a Beast attack onto a victims os. He then played the role of the victim and did not see any bad processes within the Windows Task Manager. When he checked with Black Light Security Tool, it showed the Beast program. This makes it a useful and effective tool for sniffing out and revealing rootkits.

References:

Claudiouzelac. (n.d.). rootkit.com/fuzen\_op/FU\_README.txt at master · claudiouzelac/rootkit.com. GitHub. <https://github.com/claudiouzelac/rootkit.com/blob/master/fuzen_op/FU_README.txt>

Code Red (computer worm). (2021, May 4). Wikipedia. https://en.wikipedia.org/wiki/Code\_Red\_(computer\_worm)

Computerphile. (2016b, March 2). Running a buffer overflow attack - computerphile [Video]. YouTube. <https://www.youtube.com/watch?v=1S0aBV-Waeo>

Gerencer, T. (2022, March 1). The top 10 worst computer viruses in history. HP® Tech Takes. https://www.hp.com/us-en/shop/tech-takes/top-ten-worst-computer-viruses-in-history#:~:text=Code%20Red%20%E2%80%93%20%242.4%20billion&text=Financial%20costs%20are%20pegged%20at,to%20defend%20against%20Code%20Red.

Pfleeger, C. P., Pfleeger, S. L., & Margulies, J. (2018). Security in Computing: 5th Edition. <https://dl.acm.org/citation.cfm?id=2756777>

Sanjiv Kawa. (2012, June 1). Rootkits demonstration [Video]. YouTube. https://www.youtube.com/watch?v=6CFnK20EyUU

SQL Slammer. (2021, January 21). Wikipedia. https://en.wikipedia.org/wiki/SQL\_Slammer