**Deliverable 3 – Report**

**Group 3**

Alvia Siraj (100427178)

Saad Toor (068964139)

Matteo Buonastella (102911161)

Jianpeng Zhang

Seneca College

BTN710 NAA: Information Security

Professor Navid Esfahani

July 11, 2021

Table of Contents

[Introduction 3](#_Toc76935149)

[Vulnerability 4](#_Toc76935150)

[System Setup 6](#_Toc76935151)

[Exploit 7](#_Toc76935152)

[**Format String Exploit** 7](#_Toc76935153)

[Leaking Secrets from the Stack 7](#_Toc76935154)

[Denial of Service 8](#_Toc76935155)

[**Buffer Overflow Exploit** 8](#_Toc76935156)

[Security Policy and Controls 10](#_Toc76935157)

[Conclusion 11](#_Toc76935158)

[Bibliography 12](#_Toc76935159)

# **Introduction**

In today's environment, even the smallest piece of information may be used against a company or a person. For instance, bank information, ID cards, company information, and so on. Network security is the policy, technique, and practise of preventing unauthorised access, misuse, alteration, or denial of a computer network and its resources. Policies, detection, and monitoring processes are all part of network security. This makes it extremely difficult for attackers to get information rather than simply inspecting other account credentials. Previously, Internet security was used by universities and private businesses such as NASA, Microsoft, Apple, and others. Even for the most basic issues, such as simple user identity checks, internet security has been required for a few days now.

Log in systems is well-known for their useful function of checking the user's credentials and determining if the user has access to this information. The difficult aspect is not creating a system log-in, but it is difficult to design a system that can withstand most attacks. Most individuals can construct a form using HTML, JavaScript, or any other language, but few realise that extra levels of security are required to protect the information from attackers. These extra layers offer the user with confidential comfort and build trust between the user and the customer. In this setting, secure log-in systems are extremely important since they pose dangers to larger organisations and their funds. This article discusses the following topics: vulnerability, system setup, exploitation, and security policy.

# **Vulnerability**

**Buffer Overflow Attack:**

Buffers are memory storage regions that temporarily hold data as it is transferred from one location to another. A buffer overflow or buffer overrun occurs when the volume of data exceeds the storage capacity of the memory buffer. The program that attempts to write data to the buffer overwrites neighboring memory areas. A Login Credential buffer, for example, can presume 8 bytes for username and password inputs, so the program may enter additional data beyond the buffer boundary if a transaction has a 10 bytes input (2 bytes more than expected).

As shown in the below example image:



Overwriting an application's memory via buffer overflow vulnerabilities benefits attackers. This alters the program's path, resulting in a response that destroys files or exposes sensitive data. An attacker, for example, may inject additional code and give the computer with new instructions to gain access to IT systems. If an attacker understands the memory layout of the program, he or she can deliberately feed information into the buffer that cannot hold it, overwriting the executable code sections and replacing them with their own code. For program control, an attacker may replace a pointer (an object referring to a different address in memory) with a pointer to a payload exploit. Because there are no buffer overflow safeguards in your memory, the C and C++ languages are particularly vulnerable to overwriting and reading buffer overflow attacks. For Mac OS X, Windows, and Linux, everything is written in C and C++. Included protections are applied to languages such as PERL, Java, JavaScript, and C#, reducing the possibility of buffer overflow.

There are 2 types of buffer overflow attacks:

Stack based buffer overflows: Because stack memory is used solely throughout a function, stack-based buffer overflows are more common.

Heap-based attacks are harder to execute since they necessitate overflowing the allocated memory area for a non-memory application's current runtime operation.

**Format String Attack:**

The Format String is exploited while the application examines the contents of a supplied input string. As a result, an attacker can execute code, access the stack, or cause a segmentation fault in an operating programme, resulting in unexpected behaviour that might jeopardise the system's security and stability.

Format attacks that use strings alter the application's flow. To access additional memory locations, they use library string formatting. There have been flaws when user data is sent straight to format string input for C/C++ functions (e.g., fprintf, printf, sprintf, setproctitle, syslog, ...).

The print(argv[1]) line; if you compile and run the sample program, it is vulnerable:

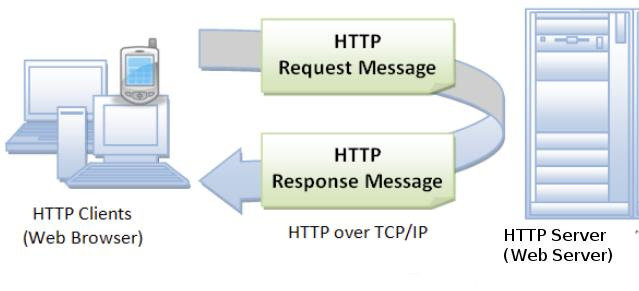
'Hello World %s % s % s % s % s % s % s % s % s % s % s % s % s % s % s % s % s %

In the next line, the printer will read all the % s in the input string as a reference to string points, with each % s attempting to interpret as a point for a string beginning at the buffer location, likely on the stack. It eventually reaches an erroneous address and attempting to access it, causing the application to crash.

# **System Setup**

To analyze the exploit as well as how to protect against it, we must first go over the system that we are using and how it will be setup. The hardware that this attack will be demonstrated on is a laptop running the Windows operating system. A web app will be used for the attack and the protocol will be an HTTP request.

The method in which an HTTP request works is that the client will make a request to the server to gain access to a specific resource that is located on the server. For example, when logging into a website you would send an HTTP request to the server to verify that your credentials (username and password) are correct and match with a user in the database. If successful, you will be “logged into” and redirected to the main portal of the website.



A potential exploit of this system is when the user inputs specific code as the username/password and the server treats the request in an unintended manor. For example, instead of entering your proper username, you enter a script that will return data from the database or cause damage to the database. This is known as an SQL injection attack. Generally speaking, the most common method used in terms of protecting yourself from these attacks is to add checks to your code to ensure that you aren’t using malicious data. From the network point of view, a common method to prevent against SQL injection attacks is to use a web application firewall (WAF). This firewall operates by monitoring the traffic on the server and searches for any patterns that it considers a threat. It acts as a border between the web app and the server and is used as a defense in depth strategy.

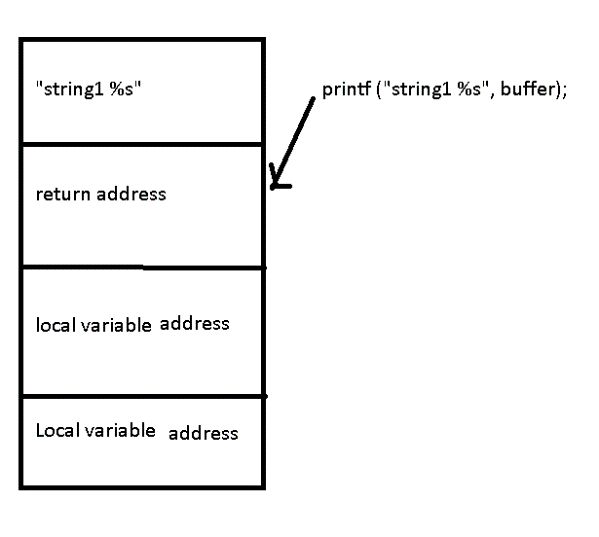
# **Exploit**

## **Format String Exploit**

An exploit is a software tool designed to take advantage of a bug or vulnerability to cause unintended or unanticipated behavior to occur on computer software, hardware or any electronic device. There are many attacks the format string vulnerabilities are susceptible to such as leaking secrets and overwriting memory addresses, and denial of service attacks. In a format string attack, the exploit works when the submitted input string is evaluated as a command by the system. This allows for the attacker to execute code and cause faults in the application that can then cause new behaviours that can affect the security of the application system. If an application uses format functions in its source-code, an attacker can explore the vulnerability by inserting formatting characters in a form of the website. If the value in a format string is controlled by the user, an attacker can exploit the syntax of the string to trigger a variety of dangerous malicious behaviour.

### Leaking Secrets from the Stack

An exploit is able to read data from memory on a format string. For Format strings that contain only format specifiers, for example **printf(“%x%n”);** for each format specifier it encounters in the format string, the printf() is then expected to find a suitable variable in its argument list. In C, the variables are stored in the stack in memory, therefore when printf() sees the format specifier it looks at the stack and reads the first value after the format string. In some programs this can include function return stack addresses, variable values, memory addresses and function parameters. This is leaking values from the stack and possibly leaking stack addresses which can then be used for buffer overflow that can be used to extract information and prepare for other attacks. Attackers can exploit these format strings to read memory from random addresses. The %n format string parameter in a format function can also be used by attackers to overwrite specific memory locations causing segmentation faults that can crash an application and denial of service attacks.



In a program designed to print an argument that receives user input without format specifiers, this leads the program to be vulnerable to an attack. The reconnaissance stage of the exploit is observing the program and how it runs. Format specifiers are added to print a string during its scanning process of the exploit stages. After adding these format specifiers the output is tested to reveal the problems and addresses being leaked. Going through such a program when it gets to printing, the address of a secret variable that contains the string is on the stack. The ultimate goal is to leak the actual string using format string vulnerability and not the address. The next stage of gaining access is when the attacker exploits the system by gaining access to the leaked memory addresses from the stack. By using format specifiers, it is easy to leak the address of the string. The signature of the attack is that the attacker now has access to important private memory from the stack.

### Denial of Service

This exploit of denial of service is caused by crashing the program using the format string. Accessing an invalid memory location from the stack to print will cause a segmentation fault within the program and crash it, not allowing one to access the program, this is a denial of service attack. Exploits can cause significant harm and damage to format string vulnerabilities.

The signature of a format string attack is that a successful exploit could lead to crash of the program or arbitrary code execution.

## **Buffer Overflow Exploit**

Buffer overflows are typically difficult to exploit but have a high likelihood. One can exploit a buffer overflow through sending data to a program that is stored in an undersized stack buffer. Information on this call stack is overwritten including the return pointer so that when the function returns it transfers control to malicious code in the attacker’s data. Buffer overflows can also lead to crashes, and lack of availability through putting a program in an infinite loop at times. Attackers often exploit buffer overflow to change the execution path of the application by overwriting the memory to take control with malicious code. This malicious code often is designed to trigger actions sending new instructions to the system that results in unauthorized access.

0 1 2 3 4 5 6 7 8 9 10

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P | A | S | S | W | 0 | R | D | 2 | 3 | 5 |

|  |
| --- |
| Function |
| Parameters |
| Return Function |
| Base Pointer |
| Buffer |

--------------------------------Buffer is 9 bytes --------------------------------------------- -- Overflow --

Malicious Code (Overwritten)

Before attack:

After Attack: after buffer overflow, actor is able to get into system and insert malicious code

Coding errors such as failure to allocate large enough buffers and not checking for overflow issues are usually the reason for buffer overflow. An example of a buffer overflow exploit is when an attacker adds malicious code into the memory stack that has been corrupted. The attacker takes advantage of the overflow. For example, if one is checking to see if a password to a program is correct, if not, no access is given to the user. However, if there is a problem or coding error within the program, this leaves it vulnerable as there is a possibility of buffer overflow. The reconnaissance stage of this exploit is observing the program and the scanning stage is understanding how it runs and where the program is most vulnerable, in this case it is in the error in coding that the variable that holds the password has a limited buffer that can be overflowed. The actual exploit in the system is when an attacker can manipulate the error in such a way that if they enter a wrong password that exceeds the buffer limit, like the diagram above, creating buffer overflow by overwriting the variable that holds the password, the incorrect password can be accepted, therefore gaining access to the server or program. Through these steps the attacker has managed to exploit buffer overflow and gain access to the root privileges of a program.

The signature of a buffer overflow attack is that it causes system crashes, and denial of service by possibly placing the system in an infinite loop or execute code on the system to bypass a security service. Buffer overflows have signatures that are triggered by the attack. There are many but a few are HTTP\_Accept\_Language\_Overflow that detects an overflow in HTTP ACCEPT field, and HTTP\_Apache\_Header\_Memory\_DoS that detect an attempt for an denial of service to a vulnerable HTTP server using a request. These are a few of the many signatures that are triggered by buffer overflow attacks.

# **Security Policy and Controls**

Firstly, whether it is to provide data for web pages or to provide software services, we need servers, so it is very important to ensure the security of servers.

Physical controls describe anything tangible that’s used to prevent or detect unauthorized access to physical areas, systems, or assets. This includes things like fences, gates, guards, security badges and access cards, biometric access controls, security lighting, CCTVs, surveillance cameras, motion sensors, fire suppression, as well as environmental controls like HVAC and humidity controls.

Technical controls (also known as logical controls) include hardware or software mechanisms used to protect assets. In computer network, by configuring a firewall, access to network communication implement measure to control the computer network, clear interviewer and data can enter into the network system, to not allow other illegal molecules. As well as data can be timely intercepted, which can effectively prevent hacking or illegal molecules into the damage from the network. Firewall, as an effective network security mechanism, has been widely used in the network system to prevent the intrusion of insecure factors in computer network to the greatest extent.

The digital envelope technology is used to ensure the confidentiality of information in electronic commerce. To ensure the integrity of e-commerce information, Hash is used as the core of digital summary technology. To ensure the validity of electronic business, information is accomplished by using digital timestamp; The digital signature technology is used to ensure that the communication in electronic commerce can not be denied. To ensure the authentication of the identity of all parties in e-commerce transactions is to establish a CA authentication system, which can give digital authentication to all parties in e-commerce transactions and must also be accompanied by security protocols. The commonly used security protocols are SSL protocol and SET protocol for secure electronic transactions. In addition, the Administrator account has the highest system permissions on the computer network, so hackers often steal the account to destroy the computer program. In order to prevent such a network threat event, you should first set a complex and strong password on the Administrator account or rename the Administrator account. Finally, you can also create an Administrator account with no administrative authority in the system to fool intruders. This will cause the intruder can not tell whether the account has the authority of the administrator, which can reduce the intruder damage to the computer network and the important information in the system.

# **Conclusion**

In conclusion, log-in systems are well-known for their useful role of checking the user's credentials and determining if the user has access to this information. The hardest part is not logging in to the system but building a system that can withstand most attacks. As previously stated, our assault is bolstered by an understanding of the following issues:

Both flaws are buffer overflow attacks (a buffer overflow or buffer overrun occurs when the data volume exceeds the memory buffer storage limit). The data-writing program overwrites neighboring memory regions in the buffer, and the format string attack (A format string attack may occur if the given data is perceived by the software as a command in an input string. By exploiting the Format string vulnerability, the attacker can execute code, access the stack, or cause a partitioning error in the running program, resulting in unexpected behavior that jeopardizes system security or stability.

To assess the exploit and how to prevent it, we must first examine the system we use and how it will be installed. This assault will be displayed on a laptop using the Windows operating system. The attack will make use of a web application, and the protocol will be HTTP.

Exploits are software tools used to make undesired or unexpected computer software, hardware, or electrical devices compatible by exploiting a fault or weakness. Numerous attacks, such as leak secrets, service rejections, exposed memory addresses, and memory overwriting, can lead to format string vulnerabilities. The exploit operates in a string attachment when the given input string is treated as a system command.

Finally, we need servers to supply data for websites or software services for security rules and controls, which implies that server security is critical. Physical checks refer to any tangible measure taken to prevent or detect unauthorized access to physical fields, systems, or property. With the understanding of these 4 topics, it is much easier to understand the process of the attack.

# **Bibliography**

9, June, and February 17. “How to Exploit Format String Vulnerabilities.” *Infosec Resources*, 28 Oct. 2020, resources.infosecinstitute.com/topic/how-to-exploit-format-string-vulnerabilities/?utm\_source=feedburner&utm\_medium=feed&utm\_campaign=Feed%3A+infosecResources+%28InfoSec+Resources%29.

A Review of Defense Against Slow HTTP Attack. (n.d.). *A Review of Defense Against Slow HTTP Attack*. <https://www.researchgate.net/figure/HTTP-Request-message-and-response-message_fig1_323194627>.

Banach, Zbigniew. “What Are Format String Vulnerabilities?” *Netsparker*, Netsparker, 7 May 2020, www.netsparker.com/blog/web-security/format-string-vulnerabilities/.

“Buffer Overflow Attacks.” *IBM*, www.ibm.com/docs/en/snips/4.6.0?topic=categories-buffer-overflow-attacks.

“Buffer Overflow Vulnerabilities, Exploits & Attacks.” *Veracode*, www.veracode.com/security/buffer-overflow.

“Buffer Overflow.” *OWASP*, owasp.org/www-community/vulnerabilities/Buffer\_Overflow.

“Format String Attack.” *Format String Software Attack | OWASP Foundation*, owasp.org/www-community/attacks/Format\_string\_attack.

Imperva. (2019, December 29). *Buffer Overflow Attack*. Learning Center. <https://www.imperva.com/learn/application-security/buffer-overflow/>.