Lab 9

Log4j Vulnerability Exploit (2021)

How did the attack happen?

The Log4j flaw, which was named CVE-2021-44228 and commonly known as "Log4Shell," was disclosed in December 2021. It exploited the Java Naming and Directory Interface (JNDI) feature of the Log4j library. Attackers could send carefully crafted input strings that, when logged by affected versions of Log4j, caused JNDI lookups to attacker-controlled servers. This enabled remote code execution (RCE) on the affected systems.

The bug was particularly dangerous because it was so easy and because Log4j was used so extensively throughout Java applications. The attackers could exploit it by injecting an attacking string into any input field logged, such as HTTP headers or user agent strings. When Log4j interpreted the string, it would execute the attacker's code.

Who were the attackers?

The vulnerability was rapidly adopted by various threat actors, such as cybercrime gangs and nation-states. Actors affiliated with China, Iran, North Korea, and Turkey employed the vulnerability in their spying and other nefarious activities, reports indicated. Ransomware gangs and botnet operators were also using the bug to spread malware and expand their networks.

What was the impact?

The Log4Shell vulnerability resulted in a global impact, hitting millions of systems across the globe in various industries like finance, healthcare, government, and technology. Such prominent organizations as Amazon, Apple, and Tesla were among those affected. The widespread nature of the vulnerability and the simplicity of exploitation led to mass scanning and attacks, with some putting over 100 million attempts at exploitation within weeks of its release.

How was it discovered?

The vulnerability was first identified by Alibaba Cloud Security Team's Chen Zhaojun on 24 November 2021. It was made public on 9 December 2021, and since then, security researchers and entities across the world began to study and mitigate

the threat. The Apache Software Foundation issued patches quickly to resolve the problem.

Why did the attack succeed?

1. Insecure JNDI Implementation: Log4j's JNDI feature allowed for dynamic lookups without proper validation, thus enabling attackers to execute arbitrary code.
2. Global Exposure of Log4j: Log4j is an at-large purpose logging library in Java applications, exposing many systems to attack.
3. Delays in Patching: Many companies waited to apply patches or didn't even know they were exposed, thus lengthening time spent in the vulnerability window.
4. Complex Dependency Chains: Log4j was deep in software dependencies, and this was hard to find and reconfigure.

How could it have been prevented?

- Secure Coding Practices: Avoid using features like JNDI lookups in logging libraries or making sure they are used securely.
- Regular Dependency Audits: Using tools to scan and update third-party libraries to the latest secure versions on a regular basis.
- Comprehensive Asset Management: Maintaining an up-to-date inventory of software components to be able to immediately detect and fix vulnerabilities.
- Proactive Monitoring: Using intrusion detection systems and monitoring logs for unusual activity to catch attempts to exploit early.

Mitigation strategies:

- Immediate Patching: Patching Log4j to version 2.17.1 or later, which resolves the vulnerability.
- Using Web Application Firewalls (WAFs): Leveraging WAFs to detect and filter out malicious input patterns associated with the exploit.
- Disabling JNDI Lookups: Disabling JNDI features if unnecessary through Log4j configuration.

- Network Segmentation: Segmentation of sensitive systems to limit the potential attack scope.
- Security Awareness Training: Educating developers and IT staff about secure coding guidelines and the importance of timely patching.

Work with a Peer

Collaborated with a peer to share findings and ensure a comprehensive understanding of the
attack and its mitigation strategies.

Reference:

Apache Log4j Security Vulnerabilities: Unit 42

CISA Apache Log4j Vulnerability Guidance: CISA

IBM's Overview of Log4j Vulnerability: IBM - United States

Research on Log4j Vulnerability and Its Severity: ResearchGate

Analysis of Initial In-The-Wild Attacks Exploiting Log4Shell: Cado Security

Topic 2: (Optional)

Twitter Bitcoin Scam (2020) – Hackers gained access to high-profile Twitter accounts and promoted a Bitcoin Scam

How did the Attack take place?

On 15 July 2020 was a day Twitter fell victim to an extensive security attack where 130 high-profile accounts were accessed by attackers. Accounts of the high-profiles Elon Musk, Barack Obama, Joe Biden, and Apple had been accessed by them. The attackers posted tweets informing that they would be giving Bitcoin away and the followers sending the Bitcoin to a particular address will receive double that which they were sending. That was the run-of-the-mill "giveaway scam."

The attack was facilitated by a spear-phishing attack on Twitter staff. Attackers had impersonated IT personnel and tricked employees into sharing their credentials, which provided the attackers with access to Twitter's internal tools and systems.

Who Were the Attackers?

The main offender was Graham Ivan Clark, who was 17 years old and from Florida. He organized the attack with others' help, including Mason Sheppard and Nima Fazeli. Clark was prosecuted as a child and sentenced to three years in a juvenile center.

Another of the main players was Joseph James O'Connor, or "PlugwalkJoe," a British citizen who was extradited to America and pleaded guilty to a string of cybercrimes associated with the hack.

What Was the Impact?

The hackers managed to get more than $118,000 in Bitcoin from unaware victims. Other than financial loss, the breach caused a grave concern for social media site security and high-profile account hacking being used to spread disinformation or create panic.

How Was It Discovered?

The scam tweets were immediately identified by Twitter users and security researchers. Twitter responded by locking the compromised accounts, removing the scam tweets, and temporarily restricting the ability of verified accounts to post new tweets while it looked into the breach.

Why Did the Attack Succeed?

1. Social Engineering: The attackers were able to successfully use social engineering tactics to compromise Twitter employees into providing access credentials.
2. Poor Internal Controls: The attack highlighted weaknesses in Twitter's internal security practices, in the form of access to admin tools.

3. Insufficient Multi-Factor Authentication (MFA): The absence of robust MFA for important internal systems allowed the attackers to gain unauthorized access.

How Could It Have Been Prevented?

- Better Employee Training: Regular training on recognizing and reacting to phishing attacks would have avoided the risk.
- Implementing MFA: Asking for MFA to reach sensitive internal tools would add an extra layer of security.
- Access Controls: Limiting access to administrative tools based on job need could reduce the attack surface.

Mitigation Strategies

- Regular Security Audits: Conducting regular audits to find and fix security vulnerabilities.
- Incident Response Planning: Developing and regularly revising incident response plans to guarantee prompt action in case of breaches.
- Monitoring and Logging: Leverage extensive monitoring to detect suspicious activity and log events for forensic analysis.

Collaborate with a Peer

Collaborated with a peer to examine the breach, analyze social engineering methods used, and determine how to enhance cybersecurity awareness and infrastructure strength.

Resources

Twitter's Official Statement on the Incident: Twitter Blogtime.com+3X Blog+3TeamPassword+3

Department of Financial Services Report: DFS Twitter ReportDepartment of Financial Services

Wikipedia Overview: 2020 Twitter Account HijackingWikipedia+1Wikipedia+1

Elliptic's Analysis on Bitcoin Laundering Post-Hack: Elliptic BlogElliptic

MITNICK Security's Breakdown: Mitnick Security BlogMitnick Security