Author: Jack Fitzgerald

Official Incident Report

Date: Mar, 07, 2024, 12:51 PM

Event ID: 235

Rule Name: SOC127 - SQL Injection Detected

Table of Contents

Alert Details

Severity: High

Type: Web Attack

Source Address: 118.194.247.28

Destination Address: 172.16.20.12

Destination Hostname: WebServer1000

Request URL: GET /?douj=3034%20AND%201%3D1%20UNION%20ALL%20SELECT%201%2CNULL%2C%27%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E%27%2Ctable_name%20FROM%20information_schema.tables%20WHERE%202%3E1--%2F%2A%2A%2F%3B%20EXEC%20xp_cmdshell%28%27cat%20..%2F..%2F..%2Fetc%2Fpasswd%27%29%23 HTTP/1.1 200 865

Device Action: Allowed

Based on the information provided in the alert, it appears that an attacker has attempted to perform an **SQL injection** attack on **WebServer1000** host **172.16.17.12**. The alert is triggered by rule SOC127 - SQL Injection Detected.

It is important to examine the attacker's proxy logs in detail.

The device action is marked **allowed**, indicating that the code injection was executed on the host machine.

Overall, it appears that the **alert** may be suspicious, and further investigation is needed to identify the extent of the alert and determine if any necessary actions are required to remediate the situation.

Detection

Verify

As a security analyst, one of the first steps we take to verify the alert and determine whether it is a **false positive** or a **true positive** incident is to analyse the logs collected from the host by our security products.

According to OWASP, an SQL injection attack consists of insertion or **injection** of a SQL query via the input data from the client to the application.

A successful SQL injection exploit can **read sensitive data** from the database, **modify database data** (Insert/Update/Delete), **execute administration operations** on the database (such as shutdown the DBMS), **recover the content of a given file** present on the DBMS file system and in some cases **issue commands** to the operating system.

SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input to affect the execution of predefined SQL commands.

Source Address: **118.194.247.28**

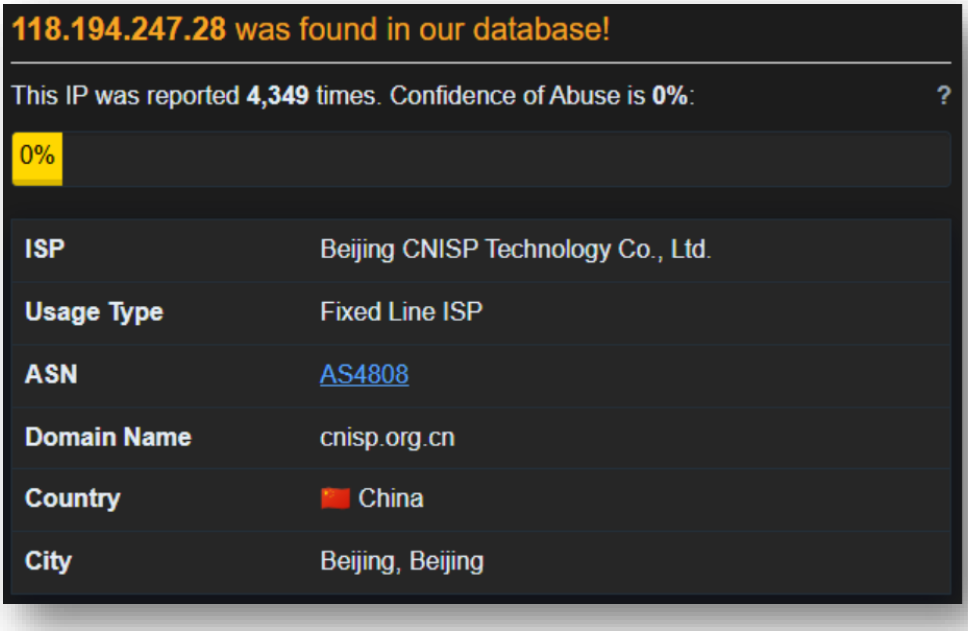
Destination Address: **172.16.20.12**

We can investigate the attacker's IP address using tools such as **VirusTotal**, **AbuseIPDB**, and **Talos Intelligence**.

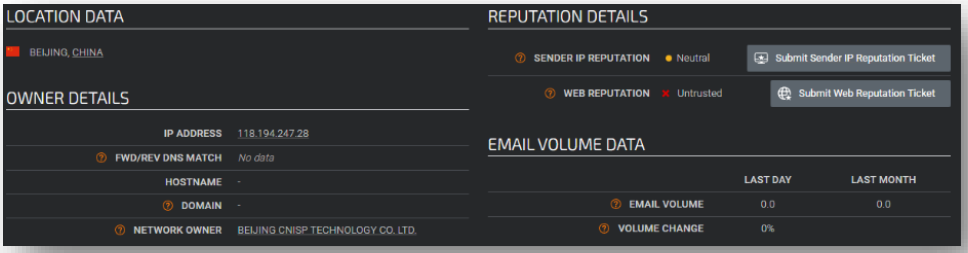
https://www.virustotal.com/gui/ip-address/118.194.247.28

7/95 security vendors flagged this IP address as malicious

https://www.abuseipdb.com/check/118.194.247.28



https://talosintelligence.com/reputation_center/lookup?search=118.194.247.28



After gathering some information about the Source IP Address, we can see:

- Ownership: External (Internet)
  - Beijing, China
  - Fixed Line ISP
  - Beijing CNISP Technology Co., Ltd.
- Domain: cnisip.org.cn
- Reputation: Malicious/Untrusted

We should examine the HTTP traffic to understand what sort of web attack is occurring.

The system received Request URL: **GET /?douj=3034%20AND%201%3D1%20UNION%20ALL%20SELECT%201%2CNULL%2C%27%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E%27%2Ctable_name%20FROM%20information_schema.tables%20WHERE%202%3E1--%2F%2A%2A%2F%3B%20EXEC%20xp_cmdshell%28%27cat%20..%2F..%2F..%2Fetc%2Fpasswd%27%29%23 HTTP/1.1 200 865**.

After decoding this GET request using **Meyerweb**, we found the following SQL injection attack.

Request URL: **GET /?douj=3034 AND 1=1 UNION ALL SELECT 1,NULL,'',table_name FROM information_schema.tables WHERE 2>1--/**/; EXEC xp_cmdshell('cat ../../../etc/passwd')# HTTP/1.1 200 865**.

Analysis

The next step of our investigation is to determine if the attacker's **SQL injection** attempt was **successful**.

Given the HTTP Response 200, and Size 865, the attacker's attempt to retrieve the **/etc/passwd** file was **successful**. The /etc/password file stores basic information for each user account, such as the **username**, **user ID**, **group ID**, and **home directory**. The existence of **xp_cmdshe**ll is another indicator that the attack was **successful**.

Containment

Based on the information gathered during the investigation, it is highly likely that the system has been compromised. Immediate isolation of the system from the network is required, and an escalation to Tier 2 is expected.

However, **WebServer1000** no longer exists on the domain, along with its IP address **172.16.20.12**.

Summary

The incident involves a compromised system named **WebServer1000** with an IP address of **172.16.17.12**. The alert was triggered by the detection of an SQL injection attempt, based on the rule SOC127 - SQL Injection Detected.

Upon further analysis, it was discovered that the source address **118.194.247.28** was **malicious**, and targeted destination address **172.16.20.12**, a **WebServer1000** on the letsdefend domain.

The SQL injection was Request URL: **GET /?douj=3034 AND 1=1 UNION ALL SELECT 1,NULL,'',table_name FROM information_schema.tables WHERE 2>1--/**/; EXEC xp_cmdshell('cat ../../../etc/passwd') HTTP/1.1 200 865**, indicating an attempt to retrieve the contents of **/etc/passwd** from **WebServer1000**.

The HTTP Response Status and Size indicate this attack was **successful**. This incident requires **Tier 2 Escalation**, and the event was identified as a **True Positive**.

Lessons Learned

- The root cause is almost always improper handling of user input
- All user-supplied data, from web forms to URL parameters and HTTP headers, should be treated as untrusted and potentially malicious
- Building SQL queries by directly concatenating user input into the query string is the primary source of SQLi vulnerabilities and should be avoided whenever possible

Remediation Actions

- Use strict allowlists to define what input is acceptable (e.g., expecting only an email or a number) rather than trying to filter out known bad characters (denylisting), which can be bypassed
- Database users should only have the minimum permissions necessary to perform their specific tasks, this limits the potential damage an attacker can cause if they successfully breach an account
- Configure applications to provide generic error messages to users, detailed database error messages can reveal valuable information about the database schema to attackers
- Ongoing security training for development teams is crucial to ensure secure coding practices are consistently applied

Appendix

MITRE ATT&CK

MITRE Tactics	MITRE Techniques
Initial Access	T1190 - Exploit Public-Facing Application
Execution	T1059 - Command and Scripting Interpreter
Credential Access	T1555 - Credentials from Password Stores

Artifacts

Value	Comment	Type
GET /?douj=3034 AND 1=1 UNION ALL SELECT 1,NULL,'',table_name FROM information_schema.tables WHERE 2>1--/**/; EXEC xp_cmdshell('cat ../../../etc/passwd') HTTP/1.1 200 865	SQL Injection Request URL	URL Address
118.194.247.28	Attacker source address	IP Address

LetsDefend Playbook

LetsDefend Event ID: 235