

BW3- Esercizio 4

Analisi del Traffico di Rete

Autore: Cybereagles

Sintesi

Il presente report documenta l'attività di analisi del comportamento dei **protocolli di rete** durante la trasmissione di dati sensibili, valutando le differenze critiche tra **comunicazioni in chiaro (HTTP)** e **cifrate (HTTPS)**. L'attività ha previsto l'intercettazione dei pacchetti grezzi e la successiva analisi forense per l'identificazione di credenziali trasmesse senza protezione rispetto a sessioni protette da protocolli crittografici.

Scopo del test e analisi dello scenario

Scenario e Obiettivi

L'attività di analisi si è svolta all'interno di un ambiente di rete controllato, basato su una distribuzione **Kali Linux**, con l'obiettivo di intercettare e dissezionare il traffico generato durante l'interazione con applicazioni web specifiche. La configurazione ha previsto l'impiego della seguente architettura:

- **Attacker: Kali Linux** (192.168.10.4) configurata sull'interfaccia *eth0* e utilizzata come stazione di sniffing.
- **Target:** Applicativo web HTTP volutamente vulnerabile ospitato all'indirizzo IP 44.228.249.3 (*testasp.vulnweb.com*).
- **Target:** Piattaforma web HTTPS ospitata all'indirizzo IP 146.75.53.91 (*netacad.com*) per la simulazione del traffico protetto.

Strumenti

- **tcpdump:** Utility da riga di comando impiegata per l'intercettazione e la registrazione dei pacchetti grezzi in transito sull'interfaccia di rete locale.
- **Wireshark:** Analizzatore di protocolli utilizzato per l'esame forense dei file di cattura, focalizzato sull'estrazione di payload in chiaro e sull'analisi dei flussi di comunicazione.

Svolgimento

Fase 1: Configurazione dell'ambiente e cattura dati HTTP

È stata identificata l'**interfaccia di rete eth0** ed è stata avviata una sessione di cattura configurata per registrare l'intero contenuto dei pacchetti. Il traffico è stato indirizzato verso un file di output locale per la successiva analisi. È stato eseguito il seguente comando a terminale:

```
$ sudo tcpdump -i eth0 -s 0 -w httpdump.pcap
```

Durante l'attività di intercettazione, si è proceduto con la navigazione verso l'ambiente volutamente vulnerabile, simulando il login tramite la form di autenticazione con username "test" e password "test". La prima sessione ha generato un **file di cattura** contenente 2972 pacchetti.

```
(kali㉿kali)-[~]
$ sudo tcpdump -i eth0 -s 0 -w httpdump.pcap
[sudo] password for kali:
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C2972 packets captured
2972 packets received by filter
0 packets dropped by kernel
```

Figura 1 Output a terminale del comando tcpdump che mostra l'avvio della cattura sull'interfaccia eth0, i parametri utilizzati e la notifica di 2972 pacchetti registrati.

Fase 2: Analisi forense del traffico HTTP in chiaro

È stata condotta un'ispezione profonda del file *httpdump.pcap* tramite **Wireshark**. Applicando il filtro di visualizzazione *http*, è stato individuato un pacchetto critico trasmesso con **metodo POST** verso la risorsa */userinfo.php*. L'analisi del payload, codificato in formato *application/x-www-form-urlencoded*, ha confermato un grave rischio di sicurezza.

È stato rilevato che il protocollo HTTP trasmette le informazioni in **formato plaintext**. Di conseguenza, è stato possibile estrarre direttamente i parametri non cifrati inseriti dall'utente, identificando la variabile **uname** con valore "test" e la variabile **pass** con valore "test".

No.	Time	Source	Destination	Protocol	Length	Info
1446	204.058798	192.168.10.4	44.228.249.3	HTTP	371	GET /style.css HTTP/1.1
1453	204.304046	44.228.249.3	192.168.10.4	HTTP	2632	HTTP/1.1 200 OK (text/css)
1455	204.304400	192.168.10.4	44.228.249.3	HTTP	431	GET /images/logo.gif HTTP/1.1
1464	204.552772	44.228.249.3	192.168.10.4	HTTP	4074	HTTP/1.1 200 OK (GIF89a)
1466	204.608882	192.168.10.4	44.228.249.3	HTTP	424	GET /favicon.ico HTTP/1.1
1472	204.855888	44.228.249.3	192.168.10.4	HTTP	1189	HTTP/1.1 200 OK (image/x-icon)
2026	460.572231	192.168.10.4	44.228.249.3	HTTP	575	POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
2028	460.821068	44.228.249.3	192.168.10.4	HTTP	330	HTTP/1.1 302 Found (text/html)
2030	460.824874	192.168.10.4	44.228.249.3	HTTP	449	GET /login.php HTTP/1.1
2032	461.076482	44.228.249.3	192.168.10.4	HTTP	2802	HTTP/1.1 200 OK (text/html)
2049	467.310523	192.168.10.4	44.228.249.3	HTTP	580	POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
2052	467.557394	44.228.249.3	192.168.10.4	HTTP	330	HTTP/1.1 302 Found (text/html)
2053	467.559695	192.168.10.4	44.228.249.3	HTTP	449	GET /login.php HTTP/1.1
2056	467.812057	44.228.249.3	192.168.10.4	HTTP	2802	HTTP/1.1 200 OK (text/html)
2071	481.318993	192.168.10.4	44.228.249.3	HTTP	578	POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
2074	481.570456	44.228.249.3	192.168.10.4	HTTP	2990	HTTP/1.1 200 OK (text/html)
2211	482.174010	192.168.10.4	216.58.209.35	OCSP	481	Request
2215	482.183347	192.168.10.4	216.58.209.35	OCSP	481	Request
2234	482.254155	216.58.209.35	192.168.10.4	OCSP	1156	Response
2242	482.261066	216.58.209.35	192.168.10.4	OCSP	1156	Response

```

▶ Frame 2071: Packet, 578 bytes on wire (4624 bits), 578 bytes captured (4624 bits)
▶ Ethernet II, Src: PCSysstemec 63:b0:05 (08:00:27:63:b0:05), Dst: 52:55:c0:a8:0a:01 (52:55:c0:a8:0a:01)
▶ Internet Protocol Version 4, Src: 192.168.10.4, Dst: 44.228.249.3
▶ Transmission Control Protocol, Src Port: 37166, Dst Port: 80, Seq: 1838, Ack: 6049, Len: 524
▶ Hypertext Transfer Protocol
▶ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▶ Form item: "uname" = "test"
  ▶ Form item: "pass" = "test"
```

Figura 2 Interfaccia di Wireshark che mostra il pacchetto HTTP POST selezionato (Frame 2071) e il dettaglio del modulo HTML espanso, con username e password visibili in chiaro nel payload.

Fase 3: Acquisizione e analisi del traffico HTTPS

La medesima metodologia è stata replicata per valutare una sessione crittografata. È stata avviata una nuova cattura di rete verso un **portale di e-learning protetto**, generando il file *httpsdump.pcap* tramite il seguente comando:

```
$ sudo tcpdump -i eth0 -s 0 -w httpsdump.pcap
```

Successivamente, il file di rete è stato esaminato in **Wireshark** impostando il filtro `tcp.port == 443` per isolare il traffico sicuro. È stato immediatamente constatato che la porzione di traffico HTTP in chiaro è stata completamente sostituita dal livello **TLS** (*Transport Layer Security*). L'ispezione dei pacchetti ha mostrato l'assenza di dati leggibili, esponendo unicamente dati crittografati etichettati come **Encrypted Application Data** e mascherando di fatto il payload tramite stringhe esadecimali.

No.	Time	Source	Destination	Protocol	Length	Info
24	0.351490	192.168.10.4	146.75.53.91	TLSv1.3	1954	Client Hello (SNI=aads.mozilla.org)
25	0.351648	146.75.53.91	192.168.10.4	TCP	60	443 → 54026 [ACK] Seq=1 Ack=1461 Win=65535 Len=0
45	0.568986	192.168.10.4	146.75.53.91	TCP	494	[TCP Retransmission] 54026 → 443 [PSH, ACK] Seq=1461 Ack=1 Win=64240 Len=440
46	0.571252	192.168.10.4	146.75.53.91	TCP	74	54032 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4056692431 TSecr=0 WS=512
47	0.571690	146.75.53.91	192.168.10.4	TCP	60	443 → 54026 [ACK] Seq=1 Ack=1901 Win=65535 Len=0
48	0.594981	146.75.53.91	192.168.10.4	TCP	60	443 → 54032 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
49	0.594720	192.168.10.4	146.75.53.91	TLSv1.3	2934	Server Hello, Change Cipher Spec, Application Data
50	0.594720	192.168.10.4	146.75.53.91	TCP	54	54032 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
51	0.595404	192.168.10.4	146.75.53.91	TCP	54	54026 → 443 [ACK] Seq=1901 Ack=2881 Win=65535 Len=0
52	0.595653	146.75.53.91	192.168.10.4	TLSv1.3	1494	Application Data, Application Data, Application Data
53	0.595658	192.168.10.4	146.75.53.91	TCP	54	54026 → 443 [ACK] Seq=1901 Ack=4321 Win=65535 Len=0
54	0.595809	146.75.53.91	192.168.10.4	TLSv1.3	265	Application Data
55	0.595811	192.168.10.4	146.75.53.91	TCP	54	54026 → 443 [ACK] Seq=1901 Ack=4532 Win=65535 Len=0
56	0.598344	192.168.10.4	146.75.53.91	TLSv1.2	276	Client Hello (SNI=firefox.settings.services.mozilla.com)
57	0.598627	192.168.10.4	146.75.53.91	TLSv1.3	118	Change Cipher Spec, Application Data
58	0.599313	146.75.53.91	192.168.10.4	TCP	60	443 → 54032 [ACK] Seq=1 Ack=223 Win=65535 Len=0
59	0.599533	146.75.53.91	192.168.10.4	TCP	60	443 → 54026 [ACK] Seq=4532 Ack=1905 Win=65535 Len=0
60	0.599577	192.168.10.4	146.75.53.91	TLSv1.3	146	Application Data
61	0.599891	146.75.53.91	192.168.10.4	TCP	60	443 → 54026 [ACK] Seq=4532 Ack=2057 Win=65535 Len=0
62	0.616065	146.75.53.91	192.168.10.4	TLSv1.3	118	Application Data

Frame 60: Packet, 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits)	0000	52 55 c0 a8 0a 01 08 00	27 63 b0 05 08 00 45 00	RU ...
Ethernet II, Src: PCSysmteec_63:b0:05 (08:00:27:63:b0:05), Dst: 52:55:c0:a8:0a:01 (52:55:c0:a8:0a:01)	0010	00 84 b8 c0 40 00 40 06	ef 60 c0 a8 0a 04 92 4b	...@
Internet Protocol Version 4, Src: 192.168.10.4, Dst: 146.75.53.91	0020	35 5b d3 0a 01 bb 25 97	e7 6a 06 58 17 b5 50 18	5[...
Transmission Control Protocol, Src Port: 54026, Dst Port: 443, Seq: 1965, Ack: 4532, Len: 92	0030	ff ff 92 c9 00 00 17 03	03 00 57 79 3c 9a 33 03
Transport Layer Security	0040	46 55 e4 b9 f8 54 1d 30	9f 5a b4 ee 08 12 62 2c	FU ...T
[Stream Index: 0]	0050	1a 53 64 1d 76 f0 13 65	64 ea 87 5f 4d 3b 45 c1	...Sd.v
TLSv1.3 Record Layer: Application Data Protocol: Hypertext Transfer Protocol	0060	af 24 ad 36 05 92 5a 92	a9 c9 c5 0a df 13 fa db	...S ...
Opaque Type: Application Data (23)	0070	42 f2 7c cc cb 47 63 ac	46 76 c1 da fe 87 f0 6e	B ...G
Version: TLS 1.2 (0x0303)	0080	9e 43 9f cb a6 d8 11 c4	13 22 3e 93 dc 12 bc b3	...C ...
Length: 87	0090	22 65		"e ...
Encrypted Application Data: 793c9a33034655e4b9f8541d309f5ab4ee0812622c1a53641d76f0136564ea875f4d3...				
[Application Data Protocol: Hypertext Transfer Protocol]				

Figura 3 Dettaglio dell'interfaccia Wireshark filtrata per la porta 443, che mostra il protocollo TLS e la sezione "Encrypted Application Data" contenente byte incomprensibili.

Fase 4: Valutazione dei benefici e dei limiti del protocollo HTTPS

Dall'analisi del comportamento dei protocolli, è stato dedotto che l'adozione del **TLS** garantisce tre pilastri fondamentali per la sicurezza: la crittazione del payload, l'integrità dei dati contro manipolazioni esterne e l'autenticazione del server tramite certificati digitali. Tali misure sono essenziali anche per la conformità normativa (es. GDPR) e per il mantenimento della fiducia degli utenti.

Tuttavia, è stato osservato che la sola presenza del **protocollo HTTPS non certifica l'assoluta affidabilità** di una piattaforma. Attori malevoli possono facilmente ottenere certificati SSL gratuiti per architettare campagne di Phishing, strutturare finte piattaforme di e-commerce (Scam) o mascherare il download di Malware, sfruttando l'indicatore visivo di sicurezza per ingannare le vittime e simulare legittimità.

Conclusioni

L'attività ha dimostrato empiricamente l'efficacia degli strumenti di sniffing per l'analisi forense e ha sottolineato l'estrema pericolosità dell'uso di protocolli non protetti come l'HTTP. Le comunicazioni non cifrate espongono i dati sensibili al rischio di intercettazione da parte di chiunque abbia visibilità sul percorso di rete. Al contrario, l'utilizzo dell'HTTPS mitiga efficacemente questo rischio, garantendo la confidenzialità attraverso l'involucro protettivo del TLS.

Si raccomandano le seguenti azioni di mitigazione:

- **Utilizzo di HTTPS:** È mandatorio assicurarsi che ogni applicazione web che gestisca processi di login o dati sensibili utilizzi certificati SSL/TLS per cifrare le transazioni.
- **Monitoraggio di Rete:** È consigliato impiegare regolarmente tool di network discovery, come **Wireshark**, all'interno della propria infrastruttura al fine di individuare e bonificare eventuali flussi di dati trasmessi in chiaro.

- **Implementazione HSTS:** È necessario configurare le policy *HTTP Strict Transport Security (HSTS)* sui server web per inibire connessioni non sicure e forzare l'esclusivo instradamento tramite canali protetti.