Overview delle vulnerabilità web

Dott. Sergio Esposito

sergio.esposito@outlook.com

Hacking Etico

- Ricercare vulnerabilità all'interno di qualsiasi sistema senza esplicita autorizzazione (scritta!) a fare ciò, può portare a delle conseguenze legali.

- Assicurarsi sempre di contrattualizzare le attività di penetration testing / vulnerability assessment o di partecipare a Bug Bounty Programs **attivi**, mantenendosi all'interno degli obiettivi, delle macchine e dei servizi *in-scope.*

HTML Injection

- A volte chiamata *«defacement».*
- Tale vulnerabilità permette di cambiare l'aspetto di una pagina iniettando del codice HTML, ad esempio all'interno di un form vulnerabile.
- Si potrebbe quindi utilizzare una pagina di un sito web vulnerabile per effettuare attacchi di **social engineering**: ad esempio si potrebbe spingere un utente a visitare un sito web malevolo, oppure ad immettere le proprie credenziali in un form di login fasullo.
- https://hackerone.com/reports/104543 con video.

CRLF Injection

- Il protocollo HTTP utilizza i caratteri **CR** e **LF** (Carriage Return e Line Feed, spesso indicati con **\r\n** o **%0d%0a**) per indicare dove finisce un header, per lasciare spazio ad un altro header o al contenuto del sito web.

- Se l'attaccante riesce ad iniettare **un** CRLF, può ingannare il browser, terminare l'header attuale ed iniziarne uno nuovo con contenuti malevoli (ad esempio con l'header Location, per reindirizzare la vittima ad un sito di phishing). Si realizza così l'**HTTP Header Injection**.

- Se l'attaccante riesce ad iniettare **due** CRLF, può ingannare il browser, terminare l'header attuale ed inserire contenuto HTML **prima** del reale sito web (talvolta, **in sostituzione** allo stesso). Si realizza così l'**HTTP Response Splitting**.

- https://hackerone.com/reports/52042 con video (Filedescriptor).

Payload Breakdown

Approfondiamo come funziona il payload utilizzato nell'esempio precedente.

- https://twitter.com/login?redirect_after_login=https://twitter.com:21/%E5%98%8A%E5%98%8Dcontent-type:text/html%E5%98%8A%E5%98%8Dlocation:%E5%98%8A%E5%98%8D%E5%98%8A%E5%98%8D%E5%98%BCsvg/onload=alert%28innerHTML%28%29%E5%98%BE

redirect_after_login - Parametro vulnerabile

%E5%98%8A - Unicode 560A - %0A - **CR**

%E5%98%8D - Unicode 560D - %0D - **LF**

%E5%98%BC - Unicode 563C - %3C - **<**

%E5%98%BE - Unicode 563E - %3E - **>**

svg/onload=alert%28innerHTML%28%29 - svg/onload=alert(innerHTML**(**)

- https://twitter.com/login?redirect_after_login=https://twitter.com:21/
  content-type:text/html
  location:

  <svg/onload=alert(innerHTML**(**)>

Cross-Site Request Forgery (CSRF)

- Da non confondere con **Cross-Site Scripting (XSS)**!
- Consiste nel fare effettuare all'utente, attraverso un sito malevolo, azioni su un altro sito nel quale l'utente è già autenticato. Spesso ciò avviene senza che l'utente ne abbia contezza.
- Spesso questi attacchi avvengono attraverso l'introduzione di tag come **** all'interno di siti malevoli, allo scopo di innescare azioni su altri siti web.
- Se Google fosse vulnerabile a CSRF, cosa farebbe il payload in grassetto?
- https://hackerone.com/reports/127703 (Zombiehelp54)

SQL Injection (SQLi)

- Forse la più famosa insieme a XSS.
- Consiste nell'iniettare codice SQL all'interno di una query per effettuare **operazioni CRUD** (**C**reate, **R**ead, **U**pdate, **D**elete) **indesiderate** all'interno di un database.
- Spesso un apice (') oppure due trattini (--) all'interno di un parametro vulnerabile permettono di aggiungere elementi indesiderati alla query, realizzando così l'attacco.
- https://hackerone.com/reports/178057 con screenshot.

Blind SQL Injection

- Letteralmente *«alla cieca»*.
- Si realizza quando un sito web è vulnerabile a SQL Injection, ma l'unico modo per estrarre dati è porre domande binarie al database ed osservare la variazione di risposta del server.
- Questo comportamento si ha ad esempio su siti vulnerabili che però sono configurati per mostrare solo errori generici.
- Una comune tecnica per vedere se è possibile una Blind SQL Injection è quella di provare ad iniettare degli *sleep()* di qualche secondo (in genere 5-10) per vedere se la pagina tarda a rispondere.
- Poniamo quindi al DB una domanda che sappiamo essere vera (ad esempio 1=1) e poi una che sappiamo essere falsa (ad esempio 1=2) e se il comportamento della pagina o del server è cambiato, possiamo provare ad estrapolare informazioni andando per tentativi.
- https://hackerone.com/reports/117073 con video.

Subdomain Takeover

- Consiste nel prendere il controllo di un sottodominio che è stato impostato per puntare ad un servizio esterno, che però non è mai stato creato o ha smesso di essere utilizzato.
- Questo può applicarsi per **Github, AWS S3, Heroku, Squarespace** e **molti altri** servizi.
- L'attaccante non fa altro che creare un account sul portale corrispondente e creare il servizio a cui punta il sottodominio, ottenendone di fatto il controllo. L'attacco è **facilissimo** da scovare e da realizzare.
- https://hackerone.com/reports/109699 con screenshot.

Remote Code Execution (RCE)

- Consiste nell'iniettare codice che viene interpretato ed eseguito da un'applicazione vulnerabile. Spesso viene associata alla **Command Injection**, in cui si iniettano comandi **alla macchina** *attraverso l'applicazione* vulnerabile (uscendo quindi dal perimetro della stessa).

- Rientra tra le vulnerabilità più critiche, in quanto ***idealmente*** permette di effettuare qualsiasi operazione sul server vittima, compreso il generare una *reverse shell*.

- https://hackerone.com/reports/212696

Approfondimento: Reverse Shell

A è dietro una connessione con NAT e ha IP interno. Non ha controllo sul suo router/firewall (ad esempio è connessa da un ufficio).

B è direttamente connesso ad Internet e ha un IP pubblico. Ha controllo sul suo router/firewall.

- **Bind Shell**
  - B richiede assistenza ad A, che deve connettersi al PC di B.
  - B lega (*bind*) cmd.exe ad una porta sul suo indirizzo IP pubblico, che comunica ad A.
  - A si connette all'indirizzo IP fornito da B ed è ora in grado di utilizzare la shell di B.
- **Reverse Shell**
  - A richiede assistenza a B, che deve connettersi al PC di A. Tuttavia A non ha modo di fornire un IP raggiungibile dall'esterno, quindi non può «bindare» la shell.
  - B resta allora in ascolto di connessioni in entrata su una porta sul suo IP pubblico, che comunica ad A.
  - A invia il controllo della propria shell verso l'IP fornito da B, che ora è in grado di usare la shell di A.

Si nota come in entrambi i casi il traffico è sempre uscente da A ed entrante in B. Lo scenario sopra è facilmente riproducibile usando **netcat**.

Server Side Request Forgery (SSRF)

- Consiste nel far effettuare ad un server vittima delle richieste HTTP per conto dell'attaccante.
- Simile al CSRF: in quel caso, ad essere attaccato era il **browser** della vittima; in questo caso, invece, la vittima è il **server**.
- Facendo fare azioni al server vittima si possono importare risorse esterne (immagini, script…) all'interno delle sue pagine web, oppure ci si può spacciare per il server vittima presso altre entità per ottenere informazioni riservate.
- https://buer.haus/2016/04/18/esea-server-side-request-forgery-and-querying-aws-meta-data/

Attacchi alla memoria
Buffer Overflow

- Si realizza quando un programma alloca una determinata quantità di memoria per scriverci sopra dei dati, ma la dimensione di tali dati **eccede la quantità di spazio allocato**, sforando in altre locazioni di memoria se la scrittura non viene arrestata.

- **Assembly, C e C++** sono notoriamente vulnerabili a buffer overflow, in quanto consentono accesso diretto alla memoria e non sono fortemente tipizzati (ovvero rischiano di gestire in maniera scorretta un type mismatch).

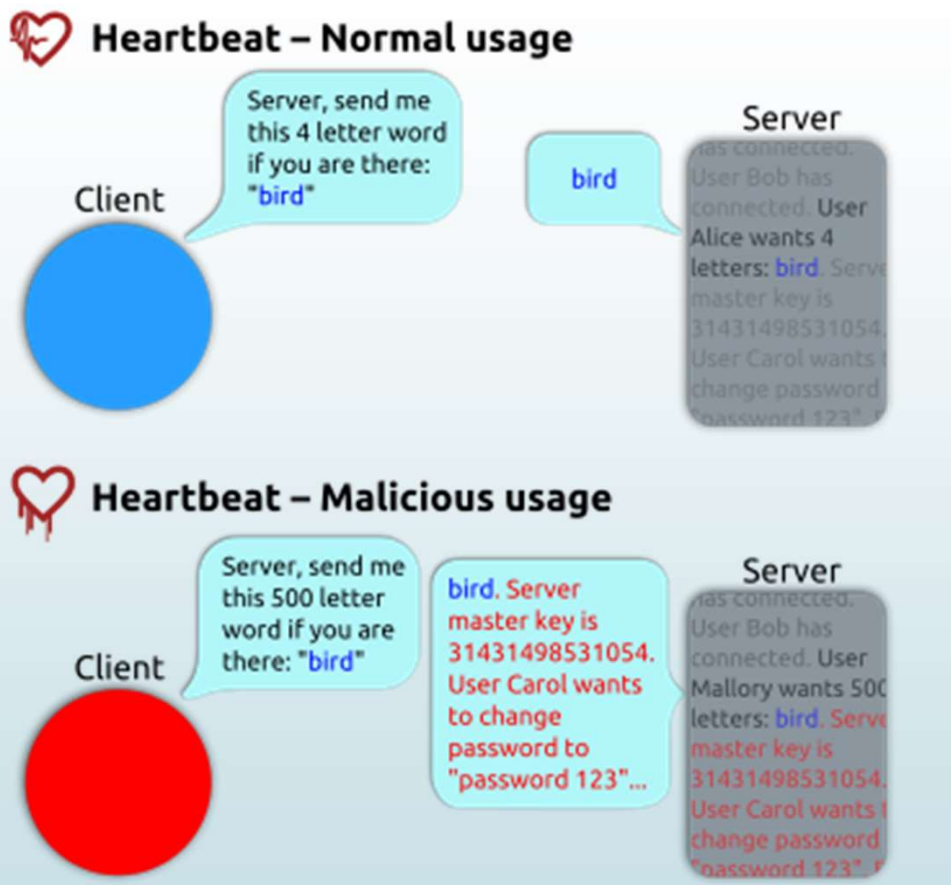
- L'attacco si realizza quando andiamo a sovrascrivere aree strategiche della memoria per ottenere comportamenti indesiderati.

- Ad esempio, riuscendo a scrivere sul **PC (Program Counter)**, possiamo dirottare il flusso di un'applicazione vulnerabile verso il payload da noi iniettato attraverso il buffer overflow.

- Per i temerari: https://hackerone.com/reports/470520 con video (e meno male!).

Attacchi alla memoria
Read out of bounds

- Mentre il buffer overflow si realizza in scrittura, il «read out of bounds» si realizza in **lettura**.
- Consiste nel leggere locazioni di memoria fuori dal perimetro a noi allocato, potenzialmente ottenendo informazioni riservate.
- **Heartbleed** è probabilmente il caso più famoso.



Attacchi alla memoria
Memory Corruption

- Consiste nel cercare di alterare il flusso di un'applicazione inserendo caratteri inaspettati o terminando inaspettatamente il flusso di byte.

- Spesso si realizza introducendo un **null byte** (%00). Il null byte viene utilizzato dai linguaggi a basso livello per indicare la fine di uno stream o di una stringa. Se inaspettato, può provocare crash o comportamenti insoliti, ad esempio l'accesso a locazioni di memoria teoricamente inaccessibili.

- Warcraft III (un noto gioco di strategia in real-time) aveva una vulnerabilità che faceva disconnettere la vittima da una partita online se le si inviava un null byte, permettendo all'attaccante di vincere quella partita senza alcuno sforzo a causa dell'abbandono dell'avversario.

Per approfondire oltre

- Esistono tanti altri tipi di attacchi!
- Molti degli esempi citati provengono dal libro **Web Hacking 101**, ottenibile gratuitamente registrandosi ad Hackerone: https://www.hackerone.com/
- Tenere sott'occhio la sezione «Hacktivity» di Hackerone è un ottimo modo per imparare nuovi attacchi: https://hackerone.com/hacktivity
- Ci sono un sacco di siti web su cui esercitarsi:
  - Damn Vulnerable Web Application (DVWA): http://www.dvwa.co.uk/
  - Hack The Box (la prima challenge è… registrarsi!): https://www.hackthebox.eu/
  - XSS Game: https://xss-game.appspot.com/level1
  - Web Goat: https://github.com/WebGoat/WebGoat/releases

Grazie per l'attenzione!

Domande?