OWASP
Open Web Application Security Project

Come mettere in sicurezza le applicazioni Legacy: un approccio pragmatico

Antonio Parata - an.parata@reply.it

Roma 12/12/2014

Reply
communication valley

Chi sono?

- Head del gruppo di R&D di Reply Communication Valley
- Un appassionato di programmazione funzionale (F#) e sviluppatore (http://nebula.tools)
- Un appassionato di software security
- Membro del board di OWASP Italy (Co-Autore della OWASP Testing Guide v2 e v3)

Reply
communication valley

Introduzione

Che cosa si intende per applicazione Legacy?

- Un'applicazione difficile da modificare/aggiornare
- Un'applicazione senza documentazione
- Un'applicazione scritta "molto tempo fa" (… in cobol)

"…to me, legacy code is simply code without tests."

Michael C. Feathers autore di **Working Effectively With Legacy Code**

Reply
communication valley

Introduzione

Perché parlare di applicazioni legacy?

Un approccio pragmatico

- Il goal è mettere in sicurezza l'applicazione e non apprendere le tecniche per comprometterla sfruttando le vulnerabilità identificate
- Dovete comunque sapere quali sono le vulnerabilità più impattanti

Reply
communication valley

Approccio

1. Eseguire attività di assessment per valutare lo stato attuale della sicurezza
2. Avviare attività mirate alla messa in sicurezza del codice
   - Non limitarsi a *fixare* soltanto le vulnerabilità identificate
3. Verifica dei progressi

Reply
communication valley

Approccio

Quali attività è consigliabile avviare per prima?

- Code Inspection
- Security Testing
- Penetration Test

Defect Removal Activities	Defects Found
Design Inspections	200
Code Inspections	400
Unit Test	125
New Function Test	75
Stress Test	25
System Test	75
Subtotal	900
Customer-reported defects(First 90 days of usage)	100
TOTAL DEFECTS FOUND	1,000

TABLE 9-22 Overview of 80 Varieties of Software Defect Removal Activities

	DEFECT REMOVAL ACTIVITIES	Number of Test Cases per FP	Defect Removal Efficiency	Bad-Fix Injection Percent
	STATIC ANALYSIS			
1.	Automated static analysis	0.00	87.00%	2.00%
2.	Requirements inspections	0.00	85.00%	6.00%
3.	External design inspection	0.00	85.00%	6.00%
4.	Use-case inspection	0.00	85.00%	4.00%
	SPECIALIZED TESTING			
51.	Virus testing	0.70	98.00%	2.00%
52.	Spyware testing	1.00	98.00%	2.00%
53.	Security testing	0.40	90.00%	4.00%
54.	Limits/capacity testing	0.50	90.00%	5.00%
55.	Penetration testing	4.00	90.00%	4.00%
56.	Reusability testing	4.00	88.00%	0.25%
57.	Firewall testing	2.00	87.00%	3.00%
58.	Performance testing	0.50	80.00%	7.00%
59.	Nationalization testing	0.30	75.00%	10.00%
60.	Scalability testing	0.40	65.00%	6.00%

Ref.Capers Jones - Software Engineering Best Practices. Lessons from Successful Projects in the Top Companies (McGraw-Hill, 2010)

OWASP Top Ten

Utilizzata per avere un'idea delle vulnerabilità più critiche

Abbastanza snella da essere letta anche dai non esperti del settore

OWASP TOP 10 – 2013
A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards

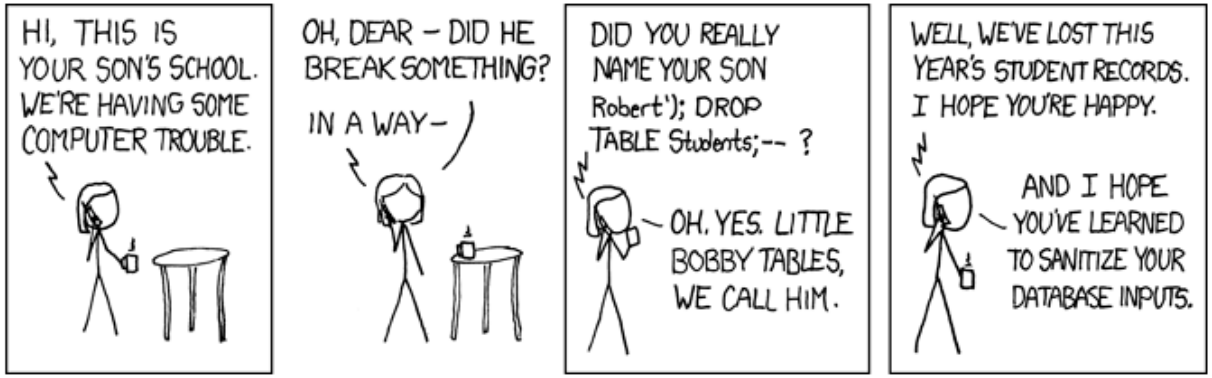
Reply
communication valley

OWASP - Proactive Controls for Developers

Fornisce una Top Ten dei controlli di sicurezza più importanti da considerare all'interno della propria applicazione

Reply
communication valley

OWASP - Proactive Controls for Developers - Parameterize Queries

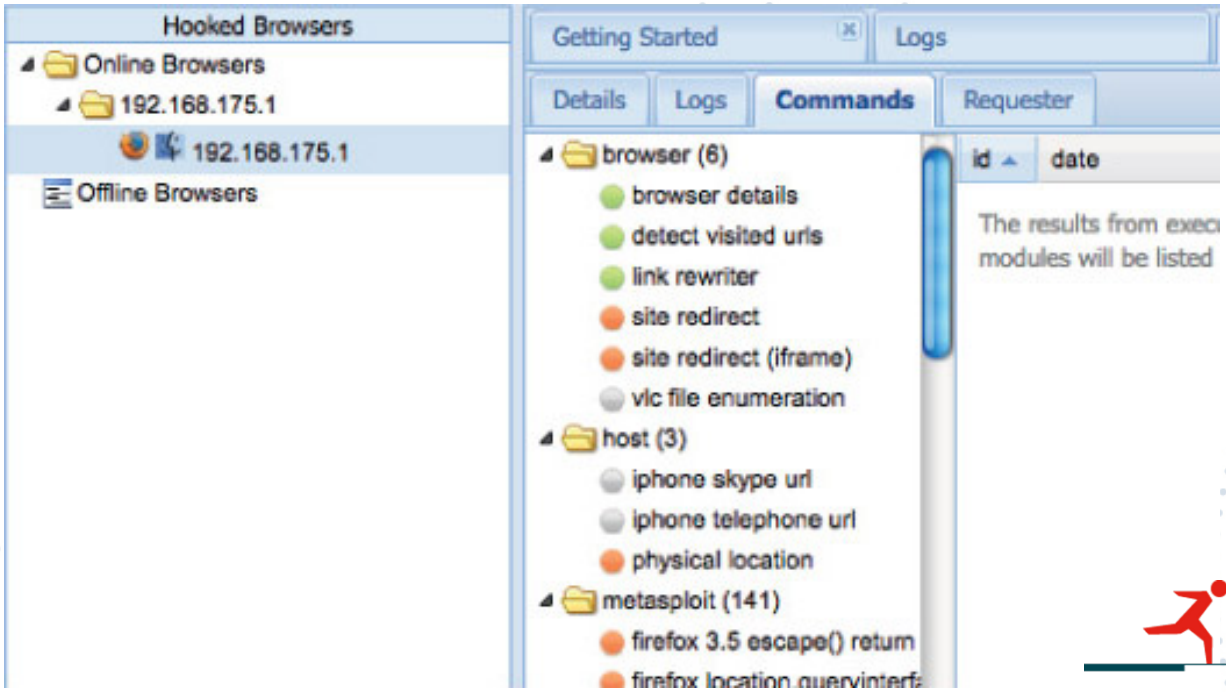



```
$stmt = $dbh->prepare("update users set
email=:new_email where id=:user_id");
$stmt->bindParam(':new_email', $email);
$stmt->bindParam(':user_id', $id);
```

Reply
communication valley

OWASP - Proactive Controls for Developers – Encode Data

Si comincia con < per finire con



Reply
communication valley

OWASP - Proactive Controls for Developers – Encode Data

La maggior parte dei Web Framework moderni hanno capability di encoding preconfigurate al loro interno

Se siete nel dubbio:

Ruby on Rails
- http://api.rubyonrails.org/classes/ERB/Util.html

Reform Project
- Java, .NET v1/v2, PHP, Python, Perl, JavaScript, Classic ASP
- https://www.owasp.org/index.php/Category:OWASP_Encoding_Project

ESAPI
- PHP.NET, Python, Classic ASP, Cold Fusion
- https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API

.NET AntiXSS Library (v4.3 NuGet released June 2, 2014)
- http://www.nuget.org/packages/AntiXss/

Reply
communication valley

OWASP - Proactive Controls for Developers – Validate All Inputs

In buona parte dei casi l'input atteso ha un formato prestabilito…

…Verificate che sia effettivamente cosi!

Approcci:

- **Whitelist** → ciò che non è permesso è rifiutato
- **Blacklist** →ciò che è malevolo è bloccato

OWASP - Proactive Controls for Developers – Implement Appropriate Access Controls

Esistenza di modelli consolidati: RBAC, ACL

Il controllo di accesso può essere abbastanza complesso da implementare, alcuni accorgimenti:

- Tutte le richieste devono passare dal controllo d'accesso
- Deny by default
- Don't reinvent the wheel

Reply
communication valley

OWASP - Proactive Controls for Developers – Establish Identity and Authentication Controls

Autenticazione è il processo che verifica che un'entità è effettivamente chi dice di essere.

Una volta autenticato tipicamente viene avviata una sessione

Assicuratevi che

- Le password siano "*saltate*" e conservate in modalità sicura (ad esempio utilizzando BCrypt)
- Il token di sessione sia debitamente protetto e non predicibile (in genere affidarsi ai meccanismi del framework sottostante è la soluzione più semplice)

Reply
communication valley

OWASP - Proactive Controls for Developers – Protect Data and Privacy

Quando inviate dati sensibili sulla rete preoccupatevi di proteggerli accuratamente

- Utilizzate HTTPS per la trasmissione di dati sensibili
- Utilizzate dei meccanismi *antitampering* per assicurare che il dato non possa essere modificato arbitrariamente

Reply
communication valley

OWASP - Proactive Controls for Developers – Implementing Loggin and Intrusion Detection

Logging non è solo per la fase di debugging

Assicuratevi di:

- Effettuare il logging di tutte le operazioni considerate sensibili (login, cambio password, …)
- Conservate i log in un "*posto sicuro*"
- Non inserite all'interno dei log informazioni sensibili (password, token di sessione, etc…)

Assicuratevi che i log siano visti o analizzati da qualcuno/ qualcosa e che in caso di problemi vengano generati degli allarmi tempestivi

Reply
communication valley

OWASP - Proactive Controls for Developers – Leverage Security Features of Frameworks and Security Libraries

A seconda del linguaggio utilizzato esistono framework che forniscono una base per la creazione dei vari meccanismi di sicurezza

Tipicamente ben scritti e con una codebase consolidata

Preoccupatevi comunque di tenerli sempre aggiornati

Reply
communication valley

OWASP - Proactive Controls for Developers – Include Security-Specific Requirements

Non è mai troppo tardi per considerare dei nuovi security requirements

Considerate:

1. Security Features and Functions
2. Business Logic Abuse Cases
3. Data Classification and Privacy Requirements

OWASP - Proactive Controls for Developers – Design and Architect Security In

In applicazioni legacy difficilmente potrà essere modificata l'architettura, cercate però di considerare

- La superficie di attacco
- Gli eventuali framework utilizzati
- Spefiche vulnerabilità derivanti dal linguaggio e/o tools utilizzati

Mi fido ma verifico

OWASP - Proactive Controls for Developers è una guida che aiuta gli sviluppatori a mettere in sicurezza il codice della propria applicazione

Serve però verificare che il codice sia stato effettivamente *messo in sicurezza*.

OWASP Application Security Verification Standard (ASVS)

Reply
communication valley

OWASP - ASVS

"The primary aim of the OWASP Application Security Verification Standard (ASVS) Project is to normalize the range in the coverage and level of rigor available in the market when it comes to performing Web application security verification using a commercially-workable open standard."

https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project

OWASP - ASVS


3
ADVANCED
2
STANDARD
1
OPPORTUNISTIC
0
CURSORY
ASVS DEFINES DETAILED VERIFICAITON REQUIREMENTS FOR LEVELS 1 AND ABOVE; WHEREAS LEVEL 0 IS MEANT TO BE FLEXIBLE AND IS CUSTOMIZED BY EACH ORGANIZATION.
OWASP ASVS LEVELS

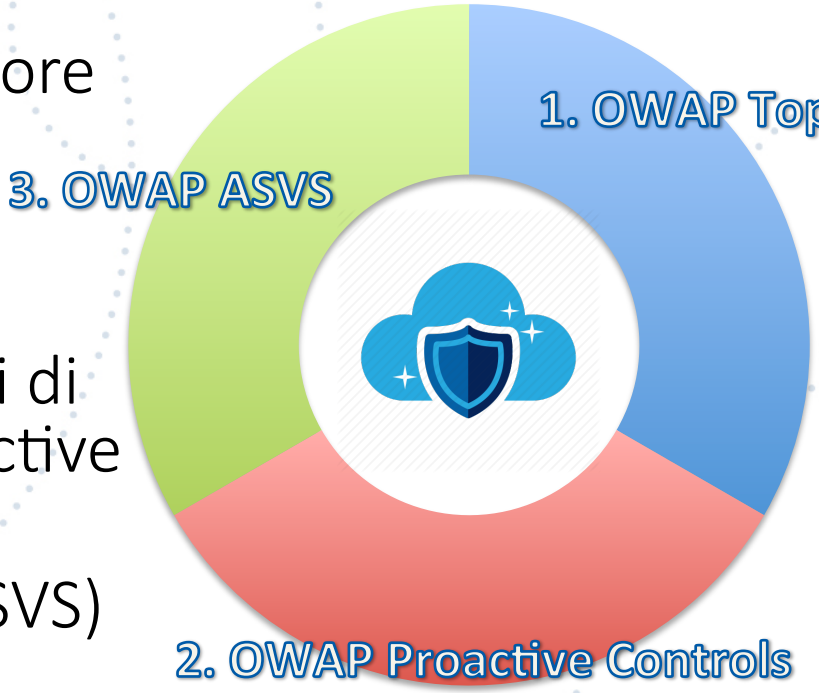
OWASP – ASVS Requisiti

V2: Authentication Verification Requirements
V3: Session Management Verification Requirements
V4: Access Control Verification Requirements
V5: Malicious Input Handling Verification Requirements
V7: Cryptography at Rest Verification Requirements
V8: Error Handling and Logging Verification Requirements
V9: Data Protection Verification Requirements
V10: Communications Security Verification Requirements
V11: HTTP Security Verification Requirements
V13: Malicious Controls Verification Requirements
V15: Business Logic Verification Requirements
V16: Files and Resources Verification Requirements
V17: Mobile Verification Requirements

Reply
communication valley

Conclusioni

1. Verificare il proprio stato eseguendo attività con valore comprovato
   - Security Testing
   - Code Inspection
2. Applicare controlli ottimali di sicurezza nel codice (Proactive Controls)
3. Verificare l'andamento (ASVS)
4. Ripetere dal punto 1 con cadenza periodica



Reply
communication valley

Q&A