Vulnerabilidades comuns da OWASP

Prof. Ricardo Sant'Ana

Descrição

Aspectos principais das vulnerabilidades comuns da OWASP, assim como suas consequências e sua exploração.

Propósito

O conhecimento das principais vulnerabilidades da OWASP, assim como o entendimento de como elas devem ser exploradas e como um código incorreto ou uma configuração inadequada pode gerar graves consequências para o ambiente de tecnologia de informação (TI), é de fundamental importância para o profissional da área de segurança da informação.

Preparação

Para realizar os experimentos, você deve realizar o download da imagem (ova) do projeto OWASP broken web application no site da OWASP e importar em uma ferramenta de virtualização, como, por exemplo, VirtualBox ou VMWare.

Objetivos

Módulo 1

Injeção, quebra de controle de acesso e falhas criptográficas

Identificar as características da injeção, da quebra de controle de acesso e das falhas criptográficas no contexto de aplicações web.

Módulo 2

Projeto inseguro, configurações incorretas e vulnerabilidade

Identificar os elementos definidores das vulnerabilidades de projeto inseguro, as configurações de segurança incorretas e a utilização de componentes vulneráveis e desatualizados.

Módulo 3

Falhas de identificação, autenticação, integridade e software

Identificar as características das vulnerabilidades de falhas de identificação e autenticação e de falhas de integridade de dados e software.

Módulo 4

Falha de registro e segurança, e falsificação de script

Identificar os elementos principais das vulnerabilidades de falha de registro e monitoração de segurança e falsificação de requisição do lado do servidor.

Introdução

O estudo e a análise das vulnerabilidades de aplicações web apresentadas no OWASP Top 10 de 2021 permitirão que o profissional da área de segurança ou de desenvolvimento aprenda sobre os principais problemas encontrados atualmente em aplicações web. Muitas vezes, desenvolvedores simplesmente criam códigos vulneráveis por eles desconhecerem as formas de exploração das vulnerabilidades utilizadas por invasores.

Iniciaremos nosso texto com uma visão geral de aplicações web, comparando essas aplicações antigas com as modernas. Em seguida, veremos as 10 vulnerabilidades apresentadas pelo OWASP, apresentando as principais características de cada uma delas.

1 - Injeção, quebra de controle de acesso e falhas criptográficas

Ao final deste módulo, você será capaz de identificar as características da injeção, da quebra de controle de acesso e das falhas criptográficas no contexto de aplicações web.

Visão geral da estrutura de uma aplicação web moderna

As aplicações web modernas utilizam tecnologias que não eram conhecidas ou amplamente utilizadas há 15 anos. A maioria dos aplicativos web era desenvolvida utilizando principalmente HTML, JavaScript e CSS.

A cada atualização de página, o cliente solicitava outra página ao servidor, deixando as aplicações web menos responsivas. O fato é que as "exigências" dos usuários por aplicações com maior usabilidade e mais responsivas aumentou.

Com isso, novas tecnologias foram desenvolvidas. Um exemplo é o emprego do Ajax, que pode ser visto como a utilização de JavaScript e XML.

Basicamente, o JavaScript é um código que executa no navegador, enquanto o uso do formato XML se justifica pela troca de informações, permitindo que sejam implementadas funcionalidades de resposta à ação do usuário de forma mais complexa.

XML

Extensible markup language.

> **Exemplo**
>
> Se você, após preencher o CEP de um formulário em uma página web, verifica que os outros campos foram preenchidos automaticamente, isso significa que o Ajax está em ação!

Muitos aplicativos web ainda podem ser representados como dois ou mais aplicativos que se comunicam ao invés de um único aplicativo monolítico. Por isso, é cada vez mais comum empregar o protocolo HTTP para disponibilizar serviços via web: os WebServices.

WebServices são serviços (ou funcionalidades) disponibilizados que utilizam tecnologias similares às das aplicações web. As **duas** mais usadas para disponibilizar WebServices são:

Simple Object Access Protocol (SOAP)

Protocolo de comunicação.

Representational State Transfer (REST)

Conjunto de princípios para transmissão de dados.

O SOAP e o REST definem padrões para comunicação entre aplicações.

Do ponto de vista de arquitetura de aplicações web, essas são as principais diferenças entre os aplicativos da web atuais e os de pouco mais de uma década atrás. Já sob um viés tecnológico, uma aplicação web moderna comum possui provavelmente o seguinte:

- API REST: Faz uso ou disponibiliza um webservice na forma de REST;
- JSON ou XML: São formas intercambiáveis de representação de dados: JSON JavaScript Object Notation e XML;

- JavaScript: Linguagem de programação executada no navegador do usuário;
- Framework para aplicações de página única (SPA), como React, Vue, EmberJS e AngularJS;
- Um sistema de autenticação e autorização;
- Um ou mais servidores web, como Apache, ExpressJS e NginX;
- Um ou mais bancos de dados, como MySQL e MongoDB;
- Armazenamento de dados do lado do cliente, como os cookies.

(SPA)

Single-page applications.

Segue agora uma revisão de algumas dessas tecnologias modernas.

REST é uma API que possui algumas características exclusivas, como o fato de ser independente da tecnologia do cliente e sem estado (de forma similar ao http). Além disso, as respostas a determinadas requisições precisam ser facilmente armazenadas em cache, permitindo, assim, maior escalabilidade. De forma simples, ele é uma especificação de arquitetura que define como as ações do HTTP (GET, POST, PUT etc.) devem ser mapeadas para recursos (endpoints e funcionalidade da API) em um servidor.

Você observou que o REST utiliza os comandos comuns do HTTP? Provavelmente você conhecia somente o GET e POST apenas, mas existem outros comandos.

Exemplo

Uma API REST é definida em um endpoint como /moderadores/cleber/logs/2020-dez. Se quisermos deletar o log de 2020-dez, basta enviarmos o comando HTTP:

DELETE /moderadores/cleber/logs/2020-dez.

Se quisermos incluir o moderador antonio, enviaremos este comando:

PUT/moderadores/antonio

Você pode verificar que, nos protocolo HTTP e nos padrões XML e JSON, ocorre as seguintes situações:

HTTP

Por meio de comandos simples já existentes no protocolo HTTP, como PUT, DELETE, GET E POST, é possível realizar um conjunto de operações em um endpoint com uma API REST.

XML e JSON

Já o XML e JSON são padrões para representar os dados utilizados, por exemplo, pelas APIs REST (não se limitando apenas a elas). Ambos são formatos intercambiáveis comumente utilizados em aplicações modernas.

Vejamos um exemplo de representação dos dados do usuário “cleber” no formato XML e JSON:

<user>
<username>cleber</username>
<password>wwdfghtvb</password>
<email>cleber@sitio.com.br</email>
</user>
	{
	"username": "cleber",
	"password": "wwdfghtvb",
	"email":"cleber@sitio.com.br"
	}

Tabela 1 : representação dos dados nos formatos XML (esquerda) e JSON (direita).
Elaborada por: Ricardo Sant'Ana.

Já os frameworks para aplicações de página única (SPA) são tecnologias utilizadas para criar sistemas web. Apesar da tradução sugerir que a aplicação terá apenas uma página, isso não é verdade.

> ### Mas o que realmente é um SPA?

Nas aplicações web antigas, as páginas são solicitadas ao servidor, e a resposta é enviada ao navegador (geralmente utilizando HTML, JavaScript e CSS). Cada nova página que precisa ser carregada se traduz em uma nova requisição para o servidor.

As SPA funcionam de maneira diferente, pois nelas não existe a necessidade de fazer requisições para o carregamento de novas páginas.

A aplicação seria, podemos dizer, "carregada" por inteiro na primeira requisição, em que todo o HTML, o CSS e o JavaScript necessários seriam carregados de uma vez. A partir desse momento, quando novas páginas precisassem ser carregadas, elas o seriam por meio da execução de código JavaScript que seleciona o novo conteúdo.

Há diversas tecnologias. Algumas delas serão apresentadas no contexto adequado.

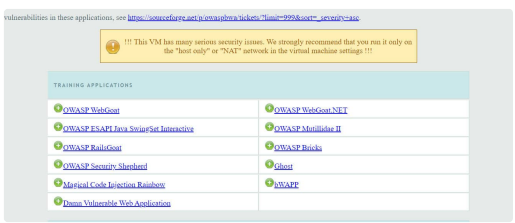
Injeção

A injeção de código é a exploração de uma falha causada pelo processamento de dados inválidos. Ela é usada por um invasor para introduzir (ou "injetar") o código em um programa vulnerável e alterar o curso de execução. O resultado de uma injeção de código bem-sucedida pode ser desastroso, podendo inclusive permitir o acesso a todos os dados existentes no banco de dados da aplicação (UTO; MELO, 2009).

Injeção de SQL

Os exemplos de injeção SQL foram testados na máquina virtual OWASP broken web application (OWASP BWA). Nesse teste, utiliza-se o endereço IP 192.168.0.53.

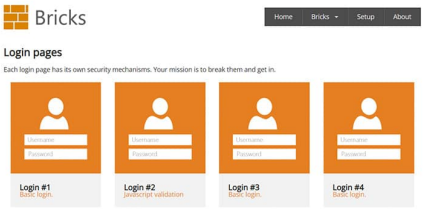
Esta imagem apresenta a página inicial do OWASP BWA:



Tela inicial da máquina virtual OWASP BWA.

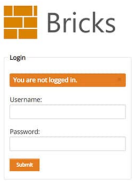
Estão disponíveis diversas aplicações de “treinamento”. Para demonstrar a injeção de SQL, utilizaremos a aplicação “**OWASP Bricks**”, desenvolvida mediante o uso da linguagem de programação PHP 5 e do gerenciador de banco de dados MySQL.

No menu da aplicação OWASP Bricks, selecionamos Bricks → Login Pages. A aplicação nos redirecionará para os desafios de login apresentados na imagem adiante.



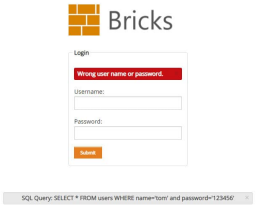
Tela dos desafios de “Login pages” da aplicação OWASP Bricks.

Selecionaremos o desafio “Login #1 Basic Login” para iniciarmos nossa apresentação. A imagem a seguir apresenta a tela inicial do desafio. O objetivo desse desafio é que você consiga realizar o login na aplicação sem saber a senha do usuário.



Desafio “Login #1 basic login”.

Testemos um par de login/senha qualquer: tente realizar a autenticação utilizando Username tom e Password 123456, como apresenta a imagem adiante.



Tentativa de autenticação com o “tom/123456”.

Após submeter os dados, a aplicação responde com “Wrong user name or password”, ou seja, não conseguimos autenticar no sistema. A aplicação Bricks apresenta o comando SQL que foi utilizado:

```
SQL
```

Claro que isso não é comum em aplicações convencionais. Tente realizar outras tentativas de autenticação utilizando outros pares de username/password e observe o comando SQL utilizado.

> **Você notou que a informação digitada no campo “Username” do formulário vai para o campo “name” do comando SQL? E que aquela digitada no campo “Password” do formulário vai para o “Password” do comando SQL?**

Observe:

```
SQL
```

```

```

É exatamente a informação de que precisávamos. Imagine o seguinte comando SQL:

```SQL

```

Note que a segunda parte da cláusula WHERE, password='' or 1='1' é sempre verdadeira, pois 1='1' → Verdadeiro, não importando o valor da password. Esse comando SQL vai buscar registros em que o name é tom, não importando o valor de password.

Mais uma vez, isso é exatamente do que precisamos. Observe agora os destaques do comando SQL:

```SQL

```

Volte para o desafio “Login #1 Basic Login” e coloque Username como **tom** e Password como '**or 1='1**. O que aconteceu? Você recebeu a mensagem “Succesfully logged in.”. **Parabéns!**

Destaquemos agora alguns pontos interessantes:

Injeção de SQL

O nome da vulnerabilidade se chama injeção de SQL pois, afinal, enviamos parte de um comando SQL no lugar da senha (' or 1='1').

Vulnerabilidade do código

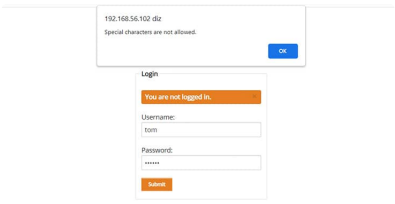
O código da aplicação web é vulnerável, porque o programador espera um comportamento **normal** do usuário, e não que ele envie algo que não seja sua senha.

Vejamos agora para o desafio “Login #2 JavaScript Validation” da aplicação Bricks. A tela inicial é muito similar à do desafio anterior.

Faça algumas tentativas para login e senha: tom/qwerty e tom/123456. Você observou alguma diferença entre as respostas?

Quando utilizamos o par tom/qwerty, a resposta é igual à resposta do desafio 1: “Wrong user name or password.”. No entanto, ao utilizarmos o par tom/123456, receberemos uma caixa de texto em nosso navegador com a mensagem: “Special characters are not allowed” (ou, em português, caracteres especiais não são permitidos). A aplicação agora está fazendo uma validação do campo password.

A imagem a seguir apresenta esta mensagem:



Mensagem de validação do campo password.

Como vamos conseguir injetar código SQL no campo password se ele não permite nada além de letras? Inicialmente, observe que a janela é resultado da execução de um programa JavaScript que verifica se o que você digitou no campo password possui apenas letras.

Dito isso, sugerimos **duas soluções** possíveis:

Desabilitar o JavaScript

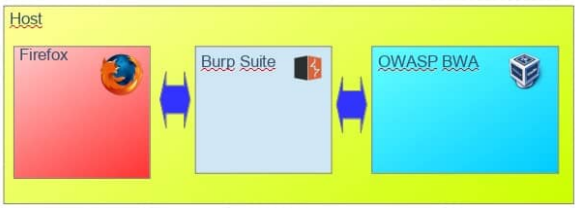
No Firefox, vá na barra de endereços e digite about:config. Aceite os riscos. Em seguida, procure por “JavaScript.enabled”.

Altere o valor para False. **Recarregue** a página do “Login #2 JavaScript Validation” e teste o par tom/123456. Você vai observar que a janela em JavaScript não vai aparecer, e sim a mensagem de “Wrong user name or password.”. Agora, coloque Username como **tom** e Password como ' **or 1='1'** da mesma forma que o desafio anterior.

Utilizar um proxy web

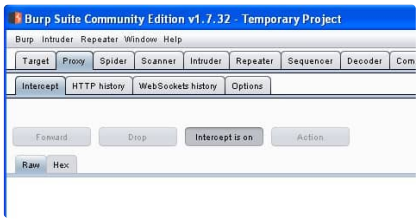
O objetivo é interceptar a requisição e alterar o valor do campo password para ' **or 1='1'** utilizando um proxy web. Um proxy web bastante conhecido e gratuito é o Burp Suite (na versão community).

A imagem adiante mostra a visão geral, após a configuração, do navegador, do Burp Suite e do servidor web. A instalação e a configuração dessa ferramenta estão fora de escopo, mas existem diversos tutoriais na internet que oferecem orientações sobre isso (UTO, 2013).



Visão geral do uso do proxy Burp Suite.

Como o Burp Suite fica no meio da comunicação, é possível configurá-lo para que ele intercepte as requisições (antes de enviá-las ao servidor), como demonstra a imagem adiante, ativando “Intercept is on” da aba Proxy.



Configuração do Burp Suite para interceptar requisições.

Utilizamos o par tom/querty e enviamos os dados pelo fato de a requisição não ser bloqueada pelo JavaScript. Na imagem adiante, verificamos que a requisição foi **interceptada**. Os dados estão na parte mais abaixo dela.

Interceptação dos dados pelo Burp Suite.

Agora, basta editar o valor de password (no Burp Suite) para ' **or 1='1'**, clicar em Forward e desabilitar o “Intercept on”. Você receberá a

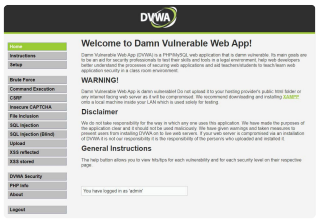
mensagem de sucesso.

Injeção de comando

Utilizaremos como exemplo outra aplicação vulnerável do OWASP BWA denominada “Damn Vulnerable Web Application” (DVWA).

Tela inicial da aplicação DVWA.

Para se autenticar na aplicação, você deve utilizar admin/admin para obter a tela com os desafios da aplicação apresentados nesta imagem:



Tela com os desafios da aplicação DVWA.

Escolhemos o desafio “Command Execution” no menu à esquerda:

Tela do desafio Command Execution.

No campo “Enter an IP address below:”, digite um número de IP válido e “submit”:

A funcionalidade recebe um IP. Realizar o comando PING apresenta o resultado no navegador do cliente. Internamente, a aplicação executa este comando:

```
TERMINAL
```

Em vermelho, destacamos o valor do campo digitado na página web. E como faremos para explorar tal funcionalidade?

Digite no campo “Enter an IP address below:” o seguinte:

```
TERMINAL
```

O resultado é que, além da saída do comando ping, o conteúdo “squeeze/sid” é apresentado.

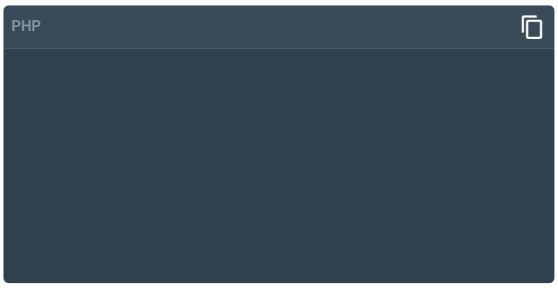
Desse modo, a injeção de comando permite a execução remota de comandos no servidor.

Analisando o código

Vamos analisar o código e entender um pouco mais o que está acontecendo.

Código do Login Pages #1 Basic Login

Eis um trecho do código da aplicação Bricks do desafio “Login Pages #1 Basic Login”:

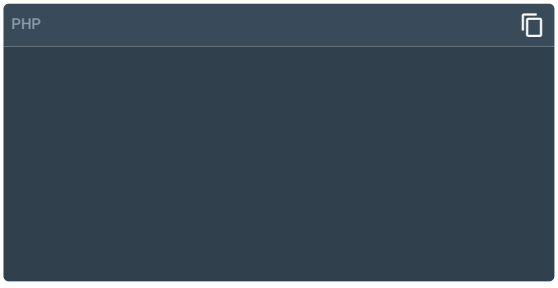


- Linha 1: O código PHP recebe via método POST o que o usuário digitou no campo username do formulário e armazena o resultado na variável $username.
- Linha 2: O código recebe via método POST o que o usuário digitou no campo passwd do formulário e armazena na variável $pwd.
- Linha 3: Ele cria uma variável $sql com o conteúdo das variáveis $username e $pwd.
- Linha 4: O comando sql é executado com mysql_query, passando como parâmetro a variável $sql.

Os dados armazenados na variável $username e $pwd são concatenados na variável $sql sem **nenhuma validação**. Por isso, podemos realizar a injeção de SQL no campo password. Uma validação de senha poderia permitir que somente números fossem utilizados e qualquer caractere diferente de número seria descartado, evitando a injeção de SQL pelo campo password.

Código do Command Execution

Observe um trecho do código do desafio de Command Execution da aplicação DVWA:



- Linha 1: O código PHP recupera o valor digitado pelo usuário no campo “Enter an IP address below:” por meio do método REQUEST e armazena na variável $target.
- Linha 2: O código PHP chama a função shell_exec, que permite executar comandos do sistema operacional via código PHP. O parâmetro passado para o shell_exe é o comando ping concatenado com $target. A saída do comando executado pelo shell_exec é armazenada na variável $cmd.

- Linha 3: A saída do comando é apresentada no navegador do usuário.

Não há validação da variável $target a permitir a realização da injeção de comandos. Como $target tem de possuir um endereço IP, o programador desse código deveria verificar se $target armazena um IP e - somente em caso positivo - realizar o shell_exec.

Injeção de SQL

Neste bate-papo, abordaremos a captura de uma prática de SQL Injection.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Quebra de controle de acessos

Para aplicações web, o controle de acesso significa colocar limites ou restrições nas seções ou nas páginas que os visitantes podem alcançar conforme suas reais necessidades.

Imagine que você tenha uma loja virtual de venda de produtos. Provavelmente, você precisará acessar a área de administração da loja para poder cadastrar novos produtos, o valor de cada item, as promoções etc.

Fica claro que o único perfil que precisa ter acesso a essas funcionalidades é quem administra o site. Por isso, deve existir alguma página de controle de acesso (login) para que os administradores da loja possam se autenticar.

> **Visitantes ou clientes não precisam ter acesso a tais funcionalidades – tampouco à página de login do administrador.**

Vamos testar? Acesse novamente a máquina OWASP BWA e procure pela aplicação "WordPress", conforme aponta na imagem a seguir.

"WordPress"

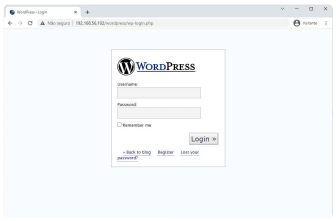
O WordPress é um sistema de gerenciamento de conteúdo gratuito. Em 2021, ele já é usado por 41,4% dos 10 milhões de sites considerados os principais na web.

Tela inicial da aplicação WordPress do OWASP BWA.

Procure nos links disponíveis o acesso à página de login de administrador do WordPress (vide a figura adiante, mas lembre-se de ajustar para o endereço IP da sua máquina de teste), que estará disponível em:

```
TERMINAL
```

Ela é a página de login do administrador e de login de usuários comuns! Se pensarmos bem, os visitantes, aqueles que só desejam consumir o conteúdo, não deveriam ter acesso a essa página.



Tela de login da aplicação WordPress da máquina OWASP BWA.

Agora, um invasor pode tentar explorar alguma vulnerabilidade, como um ataque de força bruta.

Explorando a vulnerabilidade

Vamos supor que saibamos da existência do usuário admin. Executamos o Burp Suite, acessamos a página de login e, com o "intercept on", tentamos realizar o login com o par admin/123456. Em seguida, coletamos a requisição.

```
PHP
```

Os dados a serem enviados são admin, pwd e submit. O valor do submit é constante e está codificado com "URL encode".

Ao colocar uma senha incorreta, a aplicação retorna com a mensagem "Error: incorrect password." Com essas informações, utilizamos o programa em Python modificado.

Destacaremos informações interessantes dessa requisição:

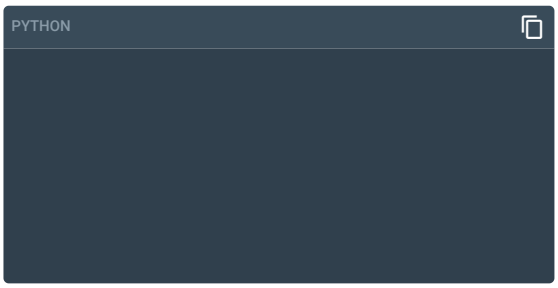
- URL de envio: http://192.168.0.53/WordPress/wp-login.php;
- Tipo de envio: POST;

- Existência de um cookie de sessão;
- Informações enviadas: log=admin&pwd=123456&submit=Login+%C2%BB&redirect_to=%2FWordPress%2Fwp-admin%2F.

Com essas informações, desenvolveremos um script em python para realizar a tarefa de força bruta de senha.

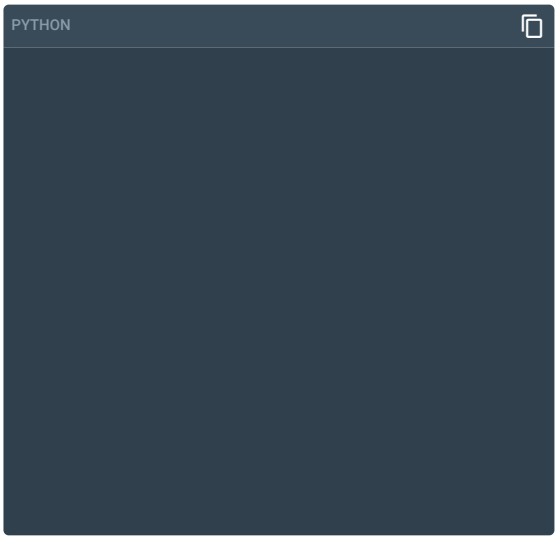
Desenvolvimento do programa

Vamos apresentar o programa em **três etapas**. Começaremos com um programa que lê o arquivo 10k-most-common.txt e apresenta a leitura de cada linha (senha):



- Linha 1: O script abre o arquivo 10k-most-common.txt.
- Linha 2: A variável “conteudo” recebe o conteúdo de todo arquivo.
- Linha 3: Fazemos um laço em que a variável “c” representa o número da linha, enquanto a variável “linha” recebe linha a linha do conteúdo.
- Linha 4: Retiramos o caractere <enter> de linha e atribuímos a variável “senha”.
- Linha 5: O valor da senha é apresentado na tela.

Agora, apresentaremos um código que envia uma requisição para a aplicação DVWA:



- Definimos um User-Agent da requisição e armazenamos na variável “headers”.
- Definimos a variável url com o conteúdo do endereço de envio.
- Definimos as variáveis user e password que serão testadas.
- Criamos um dicionário com os dados que serão enviados por post: log, pwd e submit.
- Criamos uma sessão com request.Session() e fazemos uma chamada a session.post() com os dados e url definidos. O resultado

é armazenado na variável response.

- Recuperamos o html da resposta do servidor e armazenamos na variável html. Finalmente, verificamos se a variável html contém a palavra “Incorrect Password”. Se não contém, é porque a senha utilizada está correta. Nesse caso, apresentamos o usuário e a senha na tela na linha última linha. Teste o programa trocando o valor de password para “123456”.

Agora, juntaremos as duas ideias apresentadas e criaremos um programa único que faça o ataque de força bruta:

PYTHON

O arquivo 10k-most-common.txt possui as 10 mil senhas mais comuns. Quanto maior o arquivo de senhas comuns, maiores serão nossas chances de sucesso no ataque de força bruta por dicionário. O resultado é o seguinte:

TERMINAL

Foi possível explorar a vulnerabilidade de quebra de autenticação com um ataque de força bruta, pois **a aplicação WordPress da máquina OWASP BWA possuía uma vulnerabilidade** de quebra de controle de acesso.

Exemplos de controle de acesso quebrado

Reunimos alguns exemplos de quebra de controle de acesso:

Acesso a um painel de controle/administrativo de hospedagem.

Acesso a um servidor via FTP/SFTP/SSH.

Acesso ao painel administrativo de um site.

Acesso a outros aplicativos em seu servidor.

Acesso a um banco de dados.

Ao explorar essa vulnerabilidade, o invasor pode acessar dados não autorizados, ver arquivos confidenciais ou alterar direitos de acesso. Além disso, uma aplicação com vulnerabilidades pode causar danos em outras aplicações que estão no mesmo site.

Mitigando os riscos

Podemos reduzir os riscos com algumas medidas. Apontaremos algumas delas a seguir:

Grupo de interesse

Qualquer funcionalidade do sistema deve estar restrita ao grupo de interesse.

Contas necessárias

Mantenha somente as contas necessárias.

Servidores e sites

Audite seus servidores e sites.

Ações dos usuários

Faça registro de eventos das ações dos usuários.

Multifator

Aplique a autenticação multifator.

Pontos de acesso

Desative os pontos de acesso até que sejam necessários. Reduzir o tempo de exposição dos pontos de acesso (telas de login) diminui a superfície do ataque.

Serviços

Remova serviços desnecessários.

Aplicativos

Verifique quais aplicativos precisam ser acessíveis externamente e quais não precisam.

Ambiente de produção

Tenha em mente que o ambiente de produção não deve ser usado para desenvolver ou testar atualizações.

Quebra de controle de acesso

Neste bate-papo, realizaremos a captura de uma prática de quebra de controle de acesso.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Falhas criptográficas

As falhas criptográficas ocorrem quando os dados são deixados em claro de forma acessível a qualquer pessoa ou sem o devido mecanismo de controle de acesso. Além disso, a exposição indevida pode ocorrer por conta da configuração inadequada de sistemas ou de aplicações com vulnerabilidades.

Dados confidenciais são quaisquer informações que devem ser protegidas contra acesso não autorizado. Isso pode incluir desde informações de identificação pessoal, como números de identidade ou de CPF, até informações bancárias e credenciais de login.

As falhas criptográficas e sua respectiva consequência, isto é, a exposição de dados, estão ligadas ao modo como uma organização lida com certas informações. Isso pode ocorrer por diversas formas:

Dados deixados sem criptografia em um banco de dados.

Servidor acessível a qualquer pessoa ou sem o devido mecanismo de controle de acesso.

Configuração de sistemas inadequada.

Aplicações com vulnerabilidades.

É vital para qualquer organização compreender a importância de proteger as informações e a privacidade dos usuários. No Brasil, a Lei Geral de Proteção de Dados (2018) aborda esse tema.

Podemos classificar os dados em **dois tipos**:

Dados armazenados

Todos os dados que não se movem na rede estão em repouso. Isso inclui arquivos, arquivos de backup, bancos de dados etc. Ele é considerado menos vulnerável, porém mais valioso. Um invasor pode acessá-lo, por exemplo, graças à ausência de autenticação ou devido ao controle de acesso deficiente em um repositório.

Dados em trânsito

São os dados transmitidos através de uma rede. Eles são extremamente vulneráveis, especialmente ao se moverem entre canais desprotegidos ou para as interfaces de programação de aplicações (API) que permitem que os aplicativos se comuniquem entre si.

De qualquer forma, a correta utilização de criptografia de dados sensíveis, atrelada a uma boa política de segurança, minimiza os efeitos da exposição de dados sensíveis e suas consequências.

Exemplo de aplicação de criptografia às senhas

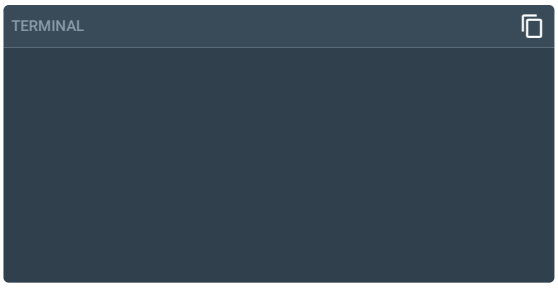
Uma forma bastante comum de se armazenar as senhas é pela aplicação de funções hash. Uma função hash é um algoritmo que mapeia dados de comprimento variável para os de comprimento fixo.

Uma de suas características é que ela é “unidirecional”, ou seja, a partir de uma informação, é possível obter o hash, embora, a partir do hash, seja extremamente complexo e computacionalmente inviável obter a informação.

Exemplo

Considere a senha q2e5t7u9. Se aplicarmos o md5 (um algoritmo comum de hash) a essa string, vamos obter 79183ceb1c094d34e19e73a427bd524c. Você mesmo pode fazer o teste!

Nas distribuições Linux, o md5sum já vem instalado. Então, para isso, basta você fazer o seguinte em um terminal:

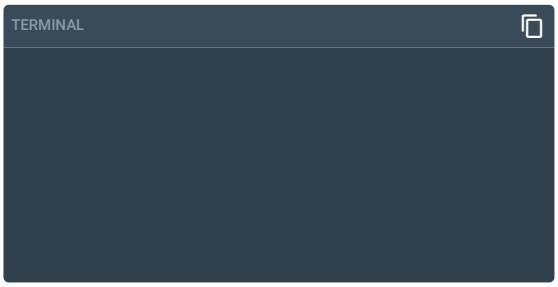


A opção -n do echo é para que não adicione um <enter> ao final da string. O resultado de echo -n "q2e5t7u9" é repassado para o comando md5sum, que calcula o hash[].

No entanto, apesar de ser complicado, a partir do hash, obter a senha de volta, os atacantes buscaram uma forma de abordar o problema: criar uma base de dados gigantesca com as principais senhas e seus respectivos hashes. Essas bases são chamadas de rainbow tables.

Cada rainbow table é construída para determinado algoritmo de hash. Uma rainbow table para senhas de até 8 caracteres ocupa a faixa de 32 petabytes[11]. Assim, se, em algum vazamento de informação, encontrarem uma lista de hashes, eles simplesmente devem realizar a busca nessas bases.

Quer um exemplo? Considere os seguintes hashes:



Acesse o site https://md5decrypt.net/en/ (em funcionamento em dezembro de 2021) e busque esses hashes. Você facilmente descobrirá que eles são os hashes para as senhas 12345678, superman e god.

Para resolver esse problema (o fato de haver rainbow tables), uma aplicação pode adicionar um "salt" na senha, isto é, uma sequência de caracteres aleatórias a uma senha. Desse modo, considerando a sequência aleatória "s$&gk%21lçp", o processo de armazenar a senha superman seria inicialmente o de concatenar as sequências de caracteres, gerando "s$&gk%21lçpsuperman", e, em seguida, aplicar o hash.

É pouco provável que o atacante construa uma base de dados com todas as sequências de 8 caracteres que ocupe 32 petabytes e que funcione apenas para o salt "s$&gk%21lçp", ou seja, para uma única aplicação.

https://md5decrypt.net/en/

Não precisa criar o link, apenas indicar um local para buscar referências de hash.

Falhas criptográficas

Neste bate-papo, realizaremos a captura de uma vulnerabilidade relacionada a falhas criptográficas.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Na página de autenticação de uma aplicação web, os campos são preenchidos e passados diretamente (sem validação) para compor o comando SQL a ser executado pelo MySQL. O usuário preencheu o campo name com tom e a password com 123456. Além disso, o comando SQL executado é este:

SELECT * FROM users WHERE name=("tom") and password= ("123456")

Qual seria o valor do campo password para que pudéssemos acessar a conta de tom sem sua senha?

A ‘ or 1=’1’

B “ or ‘=’1’

C “” or “1”=(“1

D “) or 1=(“1

E ‘) or ‘=(‘1’

Parabéns! A alternativa D está correta.

%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3Cp%20class%3D'c-paragraph'%3EParab%C3%A9ns!%20A%20alternativa%20D%20est%C3%A1%20correta.%0A%20%20%20%20%20%20%20%20%20%20

Questão 2

Determinada aplicação web não possui nenhum mecanismo para evitar o ataque de força bruta contra sua página de autenticação. Já se conhece um dos usuários do sistema: r.paul. Desejamos construir um programa em Python que realizará o ataque de força bruta contra essa aplicação web (cuja língua nativa é o inglês). Assinale a alternativa verdadeira em relação ao desenvolvimento desse programa em Python.

A Como conhecemos apenas um usuário do sistema, r.paul, não convém que o script seja desenvolvido até que se obtenha mais usuários para realizar o ataque de força bruta.

B O programa em Python utilizará uma lista de senhas mais comuns disponíveis publicamente. Quanto maior a lista, maior será o número de tentativas. Caso o usuário r.paul tenha utilizado uma senha comum, maior será a chance de descobrir sua senha.

C Como não conhecemos nada sobre o usuário r.paul, a única alternativa é tentar todas as possíveis senhas de todos os tamanhos e utilizar todos os caracteres disponíveis.

D O código para realizar o ataque de força bruta é bastante extenso e complexo, envolvendo milhares de linhas.

E Quando realizamos o envio dos dados de username e password para o servidor, o programa em Python não precisa saber o tipo de requisição (POST ou GE), pois o servidor sempre interpretará ambas.

Parabéns! A alternativa B está correta.

%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3Cp%20class%3D'c-paragraph'%3EA%20ideia%20de%20construir%20um%20programa%20em%20Python%20que%20realize%20o%20ataque%20de%2

2 - Projeto inseguro, configurações incorretas e vulnerabilidade

Ao final deste módulo, você será capaz de identificar os elementos definidores das vulnerabilidades de projeto inseguro, as configurações de segurança incorretas e a utilização de componentes vulneráveis e desatualizados.

Projeto inseguro

O projeto inseguro refere-se aos riscos associados ao projeto e à arquitetura com falhas de segurança. Esses riscos não estão vinculados a deficiências na aplicação, pois uma aplicação bem implementada ainda pode possuir vulnerabilidades de projeto ou de arquitetura potencialmente exploráveis.

Comentário

Dessa forma, você já pode concluir que não existe uma "receita de bolo" para mitigar esse tipo de problema, pois o projeto e a arquitetura de uma aplicação englobam um vasto conjunto de tecnologias. Além disso, cada caso específico precisa de uma abordagem customizada.

O projeto inseguro está relacionado à falta de controles de segurança no projeto e na arquitetura do sistema, ou seja, à falta de determinação do grau de segurança necessário para o(a) projeto/arquitetura da aplicação.

Para elucidarmos mais isso, vamos apresentar alguns exemplos de projeto inseguro:

- Não definição de limites para as entradas (inputs) da aplicação;
- Uso de funções ou APIs não seguras, como, por exemplo, a extração automática de arquivos de um pacote compactado sem levar em conta caminhos absolutos ou relativos. Um arquivo compactado pode ter o caminho relativo que sobrescreve, por exemplo, o /etc/shadow (arquivo de senhas dos usuários Linux).
- Aplicações usam privilégios elevados que não são necessários, como, por exemplo, um servidor apache executado como root.

Observe a seguir algumas formas de se prevenir o projeto inseguro:

Ciclo de vida de desenvolvimento seguro

Implementar um ciclo de vida de desenvolvimento seguro com especialistas em segurança de aplicações para avaliar a segurança do projeto e os requisitos relacionados à privacidade.

Biblioteca de padrões ou componentes seguros

Utilizar uma biblioteca de padrões de projeto seguro ou um conjunto de componentes seguros.

Métodos específicos

Aplicar métodos específicos para autenticação crítica, controle de acesso, lógica de negócios e fluxos de chaves.

Controles de segurança

Garantir que os controles de segurança façam parte das histórias de usuários (requisitos)

Verificações de front-end a back-end

Aplicar verificações de front-end a back-end, ou seja, em cada camada do sistema.

Resistência dos fluxos críticos ao modelo de ameaça

Validar a resistência dos fluxos críticos ao modelo de ameaça por meio de testes de unidade e integração.

Consumo de recursos

Restringir o consumo de recursos por usuário ou serviço.

Configurações de segurança incorretas

As configurações de segurança incorretas ocupam a 5ª posição na lista OWASP Top 10 de 2021 e são bastante comuns. Essas configurações podem causar diversos problemas em um sistema. Elencaremos alguns deles a seguir:

Falhas não corrigidas.

Configurações padrão.

Páginas não utilizadas.

Arquivos e diretórios desprotegidos.

Serviços desnecessários.

Esses problemas podem ser utilizados para realizar alguma variante do ataque de força bruta e obter acesso ao sistema.

Exemplo

Um atacante utiliza uma ferramenta para varrer todas as portas do servidor e identificar quais serviços desnecessários foram instalados.

Como a ferramenta faz isso por "tentativa e erro", chamamos tais ataques de variante de força bruta. Uma das falhas mais comuns do administrador de content management system (CMS) é manter as configurações padrão durante sua instalação. Isso acontece provavelmente por falta de conhecimento da plataforma, da rede e da programação, isto é, ela só sabe administrar o CMS.

Muitos ataques podem ser atenuados alterando as configurações padrão ao se instalar um CMS. A configuração incorreta pode acontecer em qualquer nível do sistema.

Exemplo

Serviços de rede, servidor web, servidor de aplicação, base de dados e frameworks.

Observando a vulnerabilidade

O OWASP apresenta alguns cenários possíveis:

Cenário 1

O servidor de aplicativos vem com aplicativos de exemplo que não são removidos do servidor de produção, podendo ter falhas de segurança conhecidas que os invasores usam para comprometer o servidor.

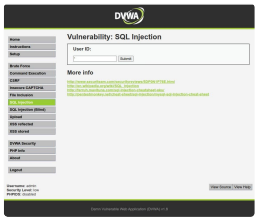
Cenário 2

A listagem do diretório não está desabilitada no servidor. Um invasor descobre que pode simplesmente listar diretórios, que encontram e baixam as classes Java compiladas. Ao realizar a engenharia reversa desse código, pode ser encontrada uma falha séria de controle de acesso.

Cenário 3

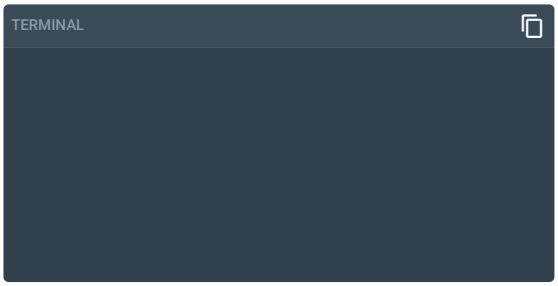
A configuração do servidor de aplicativos permite mensagens de erro detalhadas a serem devolvidas aos usuários. Isso potencialmente expõe informações confidenciais ou falhas subjacentes como versões de componentes (conhecidos por serem vulneráveis).

Para exemplificarmos o cenário 3, vamos acessar a aplicação DVWA e selecionar o desafio "SQL Injection", conforme apresentado na imagem a seguir:



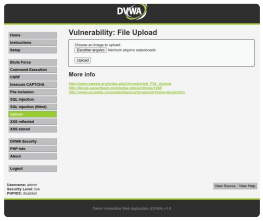
Desafio SQL Injection.

No campo “User ID:” preencha com o caracter aspas simples (‘) e clique em “Submit”. Você obterá uma mensagem de erro como a apresentada a seguir:



Além de mostrar que o campo “User ID” pode ser suscetível a uma injeção de SQL, essa mensagem também destaca o problema de uma configuração do servidor Apache/MySQL mal configurado, pois permite que mensagens de erro detalhadas do servidor sejam apresentadas para o usuário.

Outro exemplo pode ser visto no desafio Upload, conforme apresentado na imagem:



Desafio Upload.

Após análise desse desafio, é possível observar que as imagens são armazenadas no diretório “dvwa/hackable/uploads/”. Assim, acessando esse diretório via navegador, temos o seguinte resultado apresentado na imagem adiante:

Listagem do diretório “dvwa/hackable/uploads/”.

Um usuário comum não deveria ter acesso à listagem de um diretório de armazenamento de imagens. Se um atacante conseguir realizar um upload de um script php no lugar de uma imagem (essa é a forma de explorar o desafio Upload do DVWA), ele será capaz de executar o script PHP ao conhecer o caminho para ele.

Novamente, vemos que um erro de programação e outro de má configuração podem levar a sérios problemas de segurança.

Realizando varredura de vulnerabilidades

Há diversas ferramentas open-source para se realizar a varredura de vulnerabilidades.

Simples - end -->

Exemplo

Nessus, Open VAS, nmap e Nikto.

Utilizaremos o Nikto, um scanner de vulnerabilidade de linha de comando que verifica os servidores web em busca de arquivos/CGIs perigosos, softwares de servidores desatualizados e outros problemas.

Dica

Para instalar o Nikto no Ubuntu, basta realizar a atualização das listas e dos pacotes instalados. Em seguida, pode-se instalar o Nikto (como usuário root):

apt update && apt upgrade

apt -y install nikto

A utilização do Nikto é bastante simples. A partir de um terminal, execute o comando nikto -h <host>. Faremos a varredura na máquina do OWASP BWA (adapte-a ao seu ambiente):

$ nikto -h 192.168.0.53

A saída desse comando é bem extensa. Desse modo, destacaremos **dois pontos**:

- Versões antigas de software encontradas

  + PHP/5.3.2-1ubuntu4.30 appears to be outdated (current is at least 5.4.4)

  + Perl/v5.10.1 appears to be outdated (current is at least v5.14.2)

  + OpenSSL/0.9.8k appears to be outdated (current is at least 1.0.1c). OpenSSL 0.9.8r is also current.

  + mod_ssl/2.2.14 appears to be outdated (current is at least 2.8.31) (may depend on server version)

  + Apache/2.2.14 appears to be outdated (current is at least Apache/2.2.22). Apache 1.3.42 (final release) and 2.0.64 are also current.

- Encontrados diretórios de aplicações conhecidas

  + OSVDB-3092: /phpmyadmin/changelog.php: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.

  + OSVDB-3268: /test/: Directory indexing found.

  + OSVDB-3092: /test/: This might be interesting...

  + OSVDB-3092: /cgi-bin/: This might be interesting... possibly a system shell found.

  + OSVDB-3093: /.bash_history: A user's home directory may be set to the web root, the shell history was retrieved. This should not be accessible via the web.

O phpmyadmin é uma ferramenta para administrar banco de dados MySQL. Ela não deveria estar disponível para visitantes (quebra de controle de acesso). Considerando que essa ferramenta deve ser uma versão antiga, ainda pode ser possível realizar ataques de força bruta para obter acesso de administrador.

Como evitar configurações incorretas

Seguem algumas sugestões para evitar ou mitigar problemas de configurações incorretas:

Configuração de ambientes de desenvolvimento, controle de qualidade e produção

Os ambientes de desenvolvimento, controle de qualidade e produção devem ser todos configurados de forma idêntica, com credenciais diferentes usadas em cada ambiente.

Instalação mínima

Realizar sempre uma instalação mínima, sem nenhum outro recurso, componente, documentação e aplicações.

Processo de revisão e atualização

Criar um processo de revisar e atualizar as configurações.

Processo automatizado de verificação

Criar um processo automatizado para verificar a eficácia das configurações.

Configurações de segurança incorretas

Neste bate-papo, um especialista realizará a captura de uma prática de configurações de segurança incorretas.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Componentes vulneráveis e desatualizados

Atualmente, quaisquer aplicações web modernas simples, como blogs pessoais, contêm muitas dependências. Deixar de atualizar todos os softwares no back-end e no front-end de um site, sem dúvida, acarretará pesados riscos de segurança mais cedo ou mais tarde.

Exemplo

No ano de 2019, de acordo com texto publicado pelo Sucuri (2022), 56% de todos os aplicativos CMS estavam desatualizados. Além disso, grande parte dos CMS analisados continha instalações de WordPress.

O WordPress possui um ciclo de desenvolvimento seguro (SDLC) e, portanto, está constantemente atualizando seu núcleo central de código. Contudo, o mesmo não acontece com seus plug-ins, que são

desenvolvidos de forma independente por desenvolvedores terceirizados.

Esses plug-ins são o grande problema. Podemos identificar dois sem muito esforço:

(SDLC)

Software development life cycle.

A falta de atualização pelos administradores do site.

A ausência de um processo de desenvolvimento seguro para os plug-ins.

Utilizamos o CMS WordPress apenas como exemplo.

A questão que surge naturalmente é:

por que é tão problemático realizar atualizações de segurança no código?

Para respondermos a isso, ofereceremos algumas possibilidades:

Ritmo das atualizações

Desenvolvedores não conseguem acompanhar o ritmo das atualizações. A título de exemplo, no caso do núcleo central do WordPress, uma grande quantidade de programadores trabalha para seu desenvolvimento. Nos plug-ins, é comum haver 1 ou 2 desenvolvedores programando.

Código legado

O código legado não funciona em versões mais recentes de suas dependências. Exemplo: ao atualizar o núcleo do WordPress, alguns plug-ins param de funcionar.

Atualização de maneira adequada

Os administradores de sites não têm experiência para aplicar a atualização de maneira adequada.

Se você já trabalhou em algum CMS, como WordPress ou Joomla, as observações acima podem parecer um pouco exageradas; porém, sempre que a ferramenta alerta alguém sobre uma atualização no sistema e ele não a realiza, uma vulnerabilidade conhecida pode estar presente em seu CMS.

Aplicações vulneráveis

De acordo com as diretrizes OWASP, os aplicativos vulneráveis estão desatualizados nos seguintes casos:

Versões de todos os componentes

Você não conhece as versões de todos os componentes que usa (tanto do lado do cliente quanto do lado do servidor).

Software vulnerável, sem suporte ou desatualizado

O software está vulnerável, sem suporte ou desatualizado. Isso inclui o sistema operacional, o servidor web/aplicação, o sistema de gerenciamento de banco de dados (DBMS) e APIs, além de todos os componentes, ambientes de tempo de execução e bibliotecas utilizadas.

Plataforma, as estruturas e as dependências subjacentes

Você não corrige ou atualiza a plataforma, as estruturas e as dependências subjacentes de maneira oportuna e baseada em riscos. Isso geralmente acontece em ambientes em que a correção é uma tarefa mensal ou trimestral sob o controle de alterações, o que deixa as organizações vulneráveis por muitos dias ou meses.

Configuração do componentes

Você não protege as configurações dos componentes.

Prevenção

Reunimos algumas maneiras de se prevenir o uso de componentes vulneráveis:

Remover todas as dependências desnecessárias.

Obter componentes apenas de fontes oficiais.

Não usar componentes que não tenham manutenção ativa.

Usar patching virtual com a ajuda de um firewall de aplicativo de site.

Ter um inventário de todos os seus componentes do lado do cliente e do lado do servidor.

Monitorar fontes, como, (CVE) e (NVD), para obter informações sobre as vulnerabilidades nos componentes.

(CVE)

Common Vulnerabilities and Disclosures.

(NVD)

National Vulnerability Database.

Estudos de caso

Vamos analisar agora uma vulnerabilidade do CVE. Selecionamos a vulnerabilidade CVE-2021-3120, que pode ser encontrada no site do CVE.

Nessa vulnerabilidade, um invasor pode subir um código php no lugar de uma imagem. Como é possível determinar onde esse arquivo php será colocado no servidor web (caminho), o invasor poderá simplesmente executar o php.

> **O que pode ser feito com um programa em PHP? A resposta é simples: executar comandos de sistema.**

Observe o código a seguir em PHP:

```php
```

Basicamente, esse programa em PHP recebe um comando enviado ao servidor via GET e é executado no servidor via system($_GET['cmd']). A imagem adiante apresenta a tela do WebShell:

Exemplo de WebShell.

Se você digitar o comando “cat /etc/debian_version”, obterá o resultado “squeeze/sid”. Ou seja, graças a um simples programa em PHP, é possível executar comandos remotos no sistema operacional – nesse caso, o Linux. Veja como um componente vulnerável (plug-in) pode afetar toda segurança de um servidor web.

Componentes vulneráveis e desatualizados

Neste bate-papo, realizaremos a captura de uma prática sobre componentes vulneráveis e desatualizados.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Uma forma de prevenir que as vulnerabilidades de componentes conhecidos apareçam no seu CMS está mais bem apresentada em:

A Mantenha plug-ins utilizados e plug-ins não utilizados em seu ambiente.

B Mantenha componentes que não possuem manutenção, pois eles não terão vulnerabilidades.

C Atualize seu CMS e todos os plug-ins periodicamente.

D Atualize o sistema operacional apenas. Não há necessidade de atualizar a aplicação web.

E Patches de segurança para o servidor web são suficientes, pois eles já consideram a segurança de aplicações.

Parabéns! A alternativa C está correta.

%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3Cp%20class%3D'c-paragraph'%3E%C3%89%20fato%20que%20manter%20sua%20aplica%C3%A7%C3%A3o%20web%20e%20os%20respectivos%20p
ins%20em%20uso%20atualizados%20%C3%A9%20fundamental%20para%20n%C3%A3o%20deixar%20seu%20ambiente%20vulne
ins%2C%20por%20exemplo%2C%20permite%20que%20vulnerabilidades%20conhecidas%20estejam%20expostas%20a%20ataqu

Questão 2

Um exemplo de vulnerabilidade de configuração de segurança incorreta está mais bem apresentado em:

A Uma aplicação que utiliza banco de dados MySQL para gravar informações de login e senhas em texto claro.

B Uma instalação de uma aplicação que contém controles de segurança que podem ser configurados pelo usuário.

C Uma aplicação web instalada em um único servidor web sem que nenhuma outra aplicação esteja instalada no mesmo servidor.

D Uma aplicação que informa ao usuário uma mensagem de erro relacionada a um erro no banco de dados MySQL “Error: You have an error in your SQL syntax”. Esse erro está relacionado a uma entrada não fornecida pelo usuário.

E Diversas aplicações web instaladas em um mesmo servidor web.

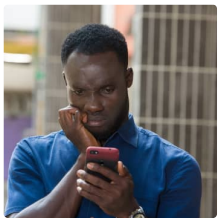
Parabéns! A alternativa D está correta.

%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3Cp%20class%3D'c-paragraph'%3EUm%20exemplo%20t%C3%ADpico%20de%20configura%C3%A7%C3%B5es%20de%20seguran%C3%A7a%20incorre

3 - Falhas de identificação, autenticação, integridade e software

Ao final deste módulo, você será capaz de identificar as características das vulnerabilidades de falhas de identificação e autenticação e de falhas de integridade de dados e software.

Falhas de identificação e autenticação



A autenticação estará “quebrada” quando o invasor for capaz de comprometer senhas, chaves ou tokens de sessão, além de informações da conta do usuário e outros detalhes, assumindo a identidade do usuário.

O problema de quebra de autenticação geralmente refere-se a problemas de lógica que ocorrem no mecanismo de autenticação do aplicativo, como, por exemplo, no gerenciamento de sessão, permitindo a realização de ataque de força bruta para adivinhar usuários ou senhas no sistema.

Descreveremos agora dois cenários típicos em que existe a vulnerabilidade de quebra de autenticação:

Cenário 1

Utilização de força bruta com o emprego de listas de senhas conhecidas. Se a aplicação web não implementar proteção automatizada contra ataques de força bruta, a aplicação poderá ser usada como um “oráculo de senha” para determinar se as credenciais são válidas, como uma rotina é implementada para testar uma lista de senhas. Se ela não funcionar, a rotina testa a próxima até que se encontre a correta.

Cenário 2

A maioria dos ataques de autenticação ocorre devido ao uso contínuo de senhas como único fator de autenticação.

A utilização da rotação de senhas e da verificação da complexidade da senha já foi considerada uma boa prática

para mitigar tal problema. No entanto, a utilização de autenticação multifator é considerada mais apropriada.

Os **três tipos** mais comuns de fatores são:

Algo que você sabe

Como uma senha ou um PIN memorizado.

Algo que você tem

Como um smartphone.

Algo que você é

Como uma impressão digital.

Atenção!

A utilização de autenticação multifator deve considerar, ao menos, dois fatores.

Realizando um ataque na prática

Vamos acessar agora a página inicial da aplicação DVWA conforme apresentamos na imagem a seguir. Temos uma tela padrão que é típica da autenticação de aplicações.

Página de Autenticação do DVWA.

Quando você tenta realizar uma autenticação e não obtém sucesso, a tela de login é recarregada com uma mensagem de “Login failed”. Precisamos descobrir as senhas de usuários. Agora, em vez de usar ferramentas prontas para realizar o ataque de força bruta, construiremos uma aplicação em Python.

Dica

Mesmo que você não tenha profundo conhecimento de Python, uma noção geral de programação vai ser suficiente para compreender os códigos e verificar como é simples realizar um ataque desse tipo.

Lista de usuários

Considere que já tenhamos obtido a lista de alguns usuários do sistema: admin, gordonb, 1337, pablo, smithy e user.

Lista de senhas

Existem diversas listas de senhas, como o Common-Credentials, que pode ser encontrado no GitHub. Utilizaremos a lista 10k-most-common.txt, que contém as 10 mil senhas mais comuns, por conveniência.

Há diversas listas de senhas comumente utilizadas. Quanto maior a lista escolhida, maior será nossa chance de descobrir a senha dos usuários.

Atenção!

Essas listas de senhas comuns serão ineficazes caso o usuário tenha escolhido uma senha forte.

Requisição de autenticação

Utilizando o Burp Suite como proxy e com a opção “Intercept on”, realizamos uma autenticação na página inicial da aplicação DVWA com o par tom/teste e coletamos a seguinte requisição:

TERMINAL

Destacaremos as informações interessantes dessa requisição:

- URL de envio: http://192.168.0.53/dvwa/login.php.
- Tipo de envio: POST.
- Existência de um cookie de sessão.
- Informações enviadas: username=tom&password=teste&Login=Login. O campo Login sempre possui o valor Login.

Desenvolvimento do programa

Vamos apresentar o programa em **três etapas**. Começaremos com um programa que lê o arquivo 10k-most-common.txt e apresenta a leitura de cada linha (senha):

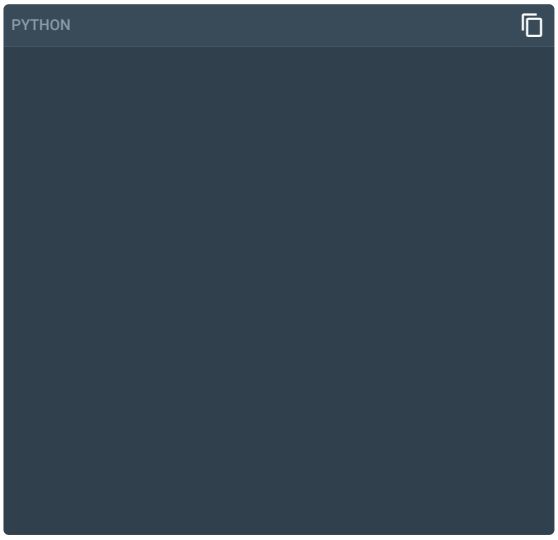
PYTHON

- Linha 1: O script abre o arquivo 10k-most-common.txt.
- Linha 2: A variável “conteudo” recebe o conteúdo de todo arquivo.
- Linha 3: Fazemos um laço em que a variável “c” representa o número da linha, enquanto a variável “linha” recebe linha a linha do conteúdo.

- Linha 4: Retiramos o caractere <enter> de linha e atribuímos a variável “senha”.
- Linha 5: O valor da senha é apresentado na tela.

Agora, apresentaremos um código que envia uma requisição para a aplicação DVWA:

PYTHON

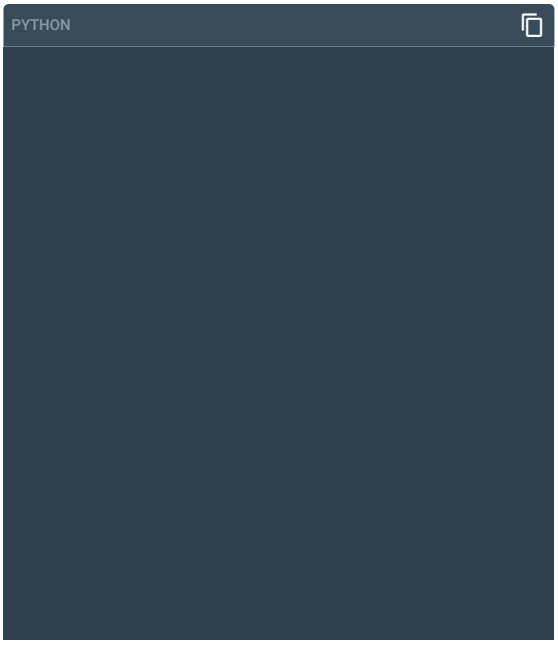


- Definimos um User-Agent da requisição e armazenamos na variável “headers”, além de definirmos a variável url com o conteúdo do endereço de envio.
- Definimos as variáveis user e password que serão testadas.
- Criamos um dicionário com os dados que serão enviados por post.
- Criamos uma sessão com request.Session() e fazemos uma chamada a session.post() com os dados e url definidos. O resultado é armazenado na variável response.
- Recuperamos o html da resposta do servidor e armazenamos na variável html.
- Verificamos se a variável html contém a palavra “failed”.

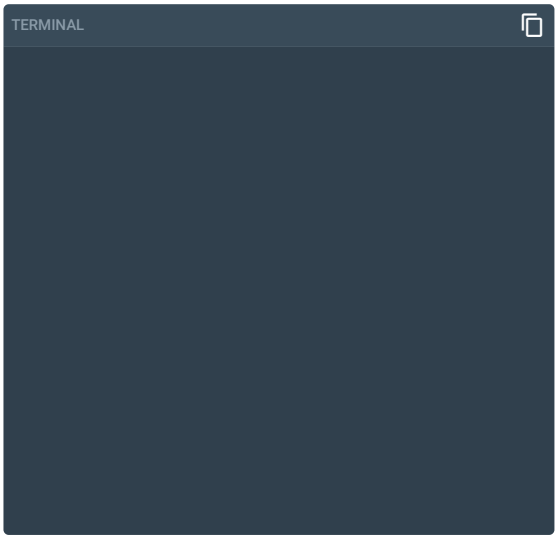
Se não contém, é porque a senha utilizada está correta. Nesse caso, apresentamos o usuário e a senha na tela na linha última linha. Teste o programa trocando o valor de password para “123456”.

Agora juntaremos as duas ideias apresentadas e criaremos um programa único que faça o ataque de força bruta:

PYTHON



O resultado desse programa será apresentado a seguir:



Conclusão

A quebra de autenticação é bastante comum, sendo fácil implementar o ataque de força bruta em páginas de autenticação vulneráveis. Essa vulnerabilidade pode ser evitada seguindo alguns conselhos:

Autenticação multifator

Implemente a autenticação multifator para evitar ataques automatizados.

Credencial padrão

Não cadastre nenhuma credencial padrão.

Verificação de senha fraca

Implemente verificações de senha fraca.

Rotação de senha

Avalie a complexidade e as políticas de rotação da senha.

Login malsucedidas

Limite ou atrase cada vez mais as tentativas de login malsucedidas.

Ataques de enumeração de contas

Garanta que os caminhos de registro, recuperação de credencial e API sejam protegidos contra ataques de enumeração de contas.

Falhas de identificação e autenticação

Neste vídeo, realizaremos uma prática sobre falhas de identificação e autenticação.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Falhas de integridade de dados e software

O ponto central da falha de integridade de dados é que, se os dados críticos usados pela aplicação não são verificados, os atacantes podem adulterá-los. Isso pode levar a problemas bastante sérios, como a introdução e a execução de um código malicioso dentro da aplicação.

Além disso, muitas aplicações incluem a funcionalidade de atualização automática de software, o que levanta preocupações sobre a integridade dos dados durante o processo de atualização. Se os invasores conseguirem realizar um ataque (MitM) e enviar um código malicioso para o aplicativo durante o processo de atualização, será muito importante que essas atualizações nunca sejam instaladas. Caso contrário, o aplicativo ficará comprometido.

(MitM)

Man in the Middle.

Exemplo

Em 2021, mais de 18 mil organizações receberam uma atualização comprometida (maliciosa) em sua instalação do SolarWind Orion. Os atacantes foram capazes de comprometer a esteira de desenvolvimento do SolarWind e introduzir um código malicioso no código original. Outra forma típica de falha de integridade de dados que você pode encontrar na prática é a desserialização insegura.

Desserialização insegura

Primeiramente, é preciso entender o que é serializar e desserializar no contexto da programação. Na ciência da computação, um objeto é como uma estrutura de dados que armazena tipos distintos de dados dentro dele, como ocorre, por exemplo, no código de uma classe em Java:

```java
```

A classe User possui **três atributos**:

Username

(string).

Logged

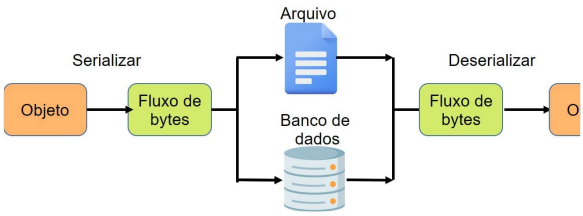
(booleano).

Admin

(booleano).

Desse modo, todos os objetos da classe User possuirão esses três atributos com valores definidos. Determinado objeto pode ter o atributo username como “carlos”, logged como False e admin como False. O funcionamento das linguagens PHP, Python e Ruby, entre outras, é similar.

A serialização é o processo de conversão de estruturas de dados complexas, como objetos e seus atributos, em um formato "mais plano" que pode ser enviado e recebido como um fluxo sequencial de bytes. Ao serializar um objeto, seu estado é persistido, isto é, os atributos do objeto são preservados para um arquivo ou banco de dados com seus valores atribuídos, conforme ilustra esta imagem:



Processo de serialização e desserialização.

A forma como os objetos são serializados depende da linguagem de programação. Algumas linguagens serializam objetos em formatos binários, como o Java, enquanto outras usam diferentes formatos de texto, como o PHP, com vários graus de legibilidade humana.

O PHP serializa os objetos na forma de arquivo texto. Observe o código adiante:

```php
```

A classe User possui três atributos: $username, $logged e $admin . Você pode ignorar o restante do código da classe User.

Criamos adiante um objeto da classe User denominado $a ao passarmos alguns parâmetros. Com eles, o objeto $a vai possuir o $username como “carlos”, o $logged como False e o $admin como False. Em seguida, mostramos na tela o resultado da serialização do objeto $a:

$username, $logged e $admin

Variáveis em PHP começam com $.

```
TERMINAL
```

Essa saída pode ser interpretada da seguinte forma:

1. O:4:"User" → um objeto com quatro caracteres chamado “User”.
2. 3 → o objeto tem três atributos.
3. s:8 → o nome do primeiro atributo contém oito caracteres e é “username”.
4. s:6 → o valor do atributo anterior contém seis caracteres e é “carlos”.
5. s:6 → o nome do segundo atributo contém seis caracteres e é “logged”.
6. b:0 → o valor do atributo anterior é booleano e é falso (zero).
7. s:5 → o nome do terceiro atributo contém cinco caracteres e é “admin”.
8. b:0 → o valor do atributo anterior é booleano e é falso (zero).

Como funciona o ataque

Quando exploramos essa vulnerabilidade, temos, de alguma forma, acesso ao objeto serializado, enquanto a desserialização será realizada pela aplicação alvo. Como o estado do objeto é persistido, o invasor pode estudar os dados serializados para identificar e editar valores de atributos interessantes. Em seguida, ele submete o objeto malicioso (adulterado) para o site que realizará o processo de desserialização.

Em termos gerais, duas abordagens podem ser adotadas ao se manipular objetos serializados:

Editar o objeto

Editar o objeto diretamente em seu formato de fluxo de bytes. Esse caso é mais comum quando o fluxo de bytes se traduz em um arquivo texto, como no caso da linguagem PHP.

Escrever um programa

Escrever um programa na linguagem em que o objeto foi criado, o que criará e serializará o novo objeto que, apesar de similar ao original, possui alguns atributos adulterados. Isso é ideal quando a linguagem realiza a serialização para formatos binários, como o Java.

Vamos supor que a aplicação tenha serializado informações de perfil do usuário conectado em um **cookie** do lado do cliente. Como explorar isso? O processo é bem simples.

Imaginemos que as informações sejam as mesmas do objeto serializado em PHP. Agora basta você **editar** o arquivo e alterar, digamos, o valor do atributo admin para True:

Pronto! Agora basta submeter esse objeto serializado para o sistema e ver se o usuário “carlos” se torna administrador. Note que a hipótese de que tornar o atributo “admin” True tornaria o usuário “carlos” um administrador parece razoável.

Um possível código vulnerável PHP que lê os dados do objeto serializado é algo como:

Esse código simplesmente realiza a desserialização do **COOKIE** e verifica se o atributo admin do objeto $user é true. Nesse caso, a exploração da vulnerabilidade seria um sucesso para o invasor.

Prevenção

Os desenvolvedores devem seguir algumas orientações para proteger o aplicativo contra falhas de integridade de dados e de software:

Sempre assine os componentes de seu aplicativo para certificar-se de que o software e os dados venham de uma fonte confiável e não foram alterados.

Certifique-se de que dependências e bibliotecas de terceiros usem apenas repositórios confiáveis.

Revise o código-fonte de seu aplicativo para evitar alterações de configuração indesejadas ou a introdução de um código malicioso no software.

Certifique-se de que nenhum dado serializado seja enviado não criptografado ou não assinado aos clientes para evitar a adulteração de dados ou ataques de repetição.

Falhas de integridade de dados e software

Neste vídeo, realizaremos uma prática sobre falhas de integridade de dados e software.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Indique a opção em que a exploração da vulnerabilidade de desserialização de objetos está mais bem descrita.

A — O invasor adiciona a um arquivo texto XML a referência a uma entidade externa de seu controle e envia tal XML para ser processado por uma aplicação que aceita esse tipo de requisição.

B — O invasor tem acesso a um objeto desserializado, como um cookie, que pode editar atributos do objeto. Assim, quando o sistema serializar este objeto adulterado, o comportamento do sistema poderá ficar alterado por causa da mudança e, assim, trazer vantagens para o invasor, como dar privilégios de administrador a um usuário do invasor.

C — Quando o objeto é desserializado para o formato binário, o invasor pode fazer um ataque de força bruta para descobrir quais são os atributos constantes nesse objeto. Com essa informação, ele pode editar os atributos e forçar um comportamento incomum no sistema.

D — O invasor força os atributos de um objeto a serem persistidos em um arquivo texto.

E — O invasor cria um programa para serializar um objeto persistido em arquivo e, com isso, obtém as credenciais de acesso do sistema.

Parabéns! A alternativa B está correta.

%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3Cp%20class%3D'c-paragraph'%3EA%20vulnerabilidade%20de%20desserializa%C3%A7%C3%A3o%20de%20objetos%20ocorre%20quando%20o%20in

Questão 2

Qual das afirmações a seguir descreve melhor um exemplo de falha de identificação e autenticação?

A — Em uma página web de autenticação, um usuário fornece seu nome de usuário e senha. Ao acessar o sistema, ele percebe que tem acesso a mais funcionalidades do que deveria.

B — Há o uso de um mecanismo de autenticação multifator, como, por exemplo, uma senha e uma impressão digital (biometria) em determinada aplicação.

C — A aplicação implementa um mecanismo que, após um erro em uma tentativa de falha de autenticação, espera 5 segundos antes de solicitar novamente o nome do usuário ou a senha.

D — A aplicação permite diversas tentativas consecutivas de autenticação, enquanto os usuários utilizam senhas comumente empregadas.

E — Duas aplicações web utilizam o mesmo mecanismo de autenticação.

Parabéns! A alternativa B está correta.

%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3Cp%20class%3D'c-paragraph'%3E%C3%89%20comum%20haver%20ataques%20de%20for%C3%A7a%20bruta%20de%20autentica%C3%A7%C3%A3o

4 - Falha de registro e segurança, e falsificação de script

Ao final deste módulo, você será capaz de identificar os elementos principais das vulnerabilidades de falha de registro e monitoração de segurança e falsificação de requisição do lado do servidor.

Falha de registro e monitoração de segurança

O problema da falta de registro (registro de eventos ou logs) e de monitoração é algo sistemático, ou seja, envolve todas as camadas da arquitetura do seu sistema de informação.

Desse modo, a importância de se proteger todo ecossistema que envolve um site não pode ser subestimada, pois, ainda que não seja possível obter 100% de segurança, existem maneiras de manter seu site monitorado regularmente para que, quando uma invasão ocorrer, você possa tomar medidas imediatas.

Não ter um processo eficiente de registro de eventos e monitoramento pode aumentar os danos de comprometimento de um site. O registro de ambos pode ser separado em três partes:

Coleta de registros

Inclui o enriquecimento de registros, como, por exemplo, análise, conversão e filtragem.

Gerenciamento de log

Mantém políticas de retenção de dados e índices para melhor desempenho, implementa controle de acesso (afinal, os logs contêm informações confidenciais) etc.

Monitoramento/análise de log

Visualização, alerta e relatórios.

Estudos de caso e causas

O próprio OWASP apresenta alguns cenários em que a falta de registo de eventos e/ou de monitoramento pode trazer problemas:

Cenário 1

Um software de fórum de projeto de código aberto executado por uma pequena equipe foi hackeado usando uma falha em seu software. Os invasores conseguiram limpar o repositório de código-fonte interno que contém a próxima versão e todo o conteúdo do fórum. Embora a fonte pudesse ser recuperada, a falta de monitoramento, de registro ou de alerta levou a uma violação muito pior. O projeto de software do fórum não está mais ativo como resultado desse problema.

Cenário 2

Um invasor procura usuários com uma senha comum. Eles podem assumir todas as contas com essa senha. Para todos os outros usuários, tal varredura deixa apenas um login falso para trás. Após alguns dias, isso pode ser repetido com uma senha diferente.

Cenário 3

Um grande varejista dos EUA supostamente tinha uma caixa de proteção interna de análise de malware analisando anexos. O software sandbox detectou um software potencialmente indesejado, mas ninguém respondeu a essa detecção. O sandbox vinha produzindo avisos por algum tempo antes de detectar a violação devido a transações fraudulentas com cartão por um banco externo.

Mas quais seriam as causas do registro de monitoramento insuficientes? A lista a seguir não é completa, mas se trata de um indicativo de quais seriam esses motivos:

Falta de registro de eventos

Por exemplo, eventos de falha de autenticação ou eventos/transações de críticas.

A falta de backup dos registros

Um invasor poderia simplesmente apagar seus rastros editando os arquivos de registro de eventos.

Configurações incorretas no software de monitoramento

Para que os alertas sobre eventos importantes não sejam mostrados.

Informações do registro de eventos

São insuficientes para que uma equipe forense possa analisar o acontecido.

Ferramentas OpenSource

Existem diversas ferramentas e frameworks de detecção de intrusão de código aberto que podem ajudá-lo a automatizar o monitoramento do seu sistema. Destacaremos algumas delas adiante:

Nagios

Fornece gerenciamento completo e monitoramento de logs de aplicativos, arquivos de log, logs de eventos, logs de serviço e logs de sistema em servidores Windows, Linux e Unix, além de alertá-lo quando os padrões de log forem detectados.

Snort

Ferramenta de detecção e prevenção de intrusão que pode realizar a análise de tráfego em tempo real e o registro de pacotes em redes.

Splunk

Consolida dados de log e de máquina, incluindo logs de aplicativos. Você pode coletar, armazenar, indexar, pesquisar, correlacionar, visualizar, analisar e relatar qualquer dado gerado por um serviço.

OSSEC

Trata-se de um sistema de detecção de intrusão baseado em host de código aberto que realiza análise de log, verificação de integridade de arquivo, monitoramento de política, detecção de rootkit, alerta em tempo real e respostas ativas.

Tripwire

Ferramenta leve de segurança e integridade de dados, o Tripwire é útil para monitorar e alertar sobre alterações de arquivos específicos em servidores Linux.

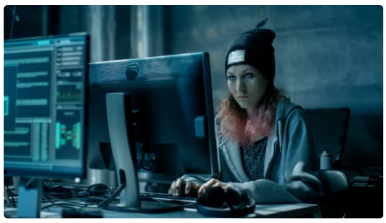
Falha de registro e monitorização de segurança

Neste vídeo, realizaremos a captura de uma prática sobre falha de registro e monitorização de segurança.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Falsificação de requisição do lado do servidor

Visão geral



(SSRF) ou falsificação de requisição do lado do servidor é um ataque que força um usuário final a executar ações indesejadas em um aplicativo da web no qual ele esteja autenticado no momento. Em um primeiro instante, isso pode parecer um tanto quanto complicado, mas, na realidade, não é.

(SSRF)

Server-Side Request Forgery.

Imagine que determinada aplicação em que você, vítima, está autenticado tenha uma funcionalidade de troca de senha. Obviamente, essa troca só é executada para usuários autenticados.

Suponha que, ao realizar a troca da senha, uma requisição do tipo GET seja enviada ao servidor com o seguinte conteúdo:

```
TERMINAL
```

Nessa requisição, podemos identificar os campos fornecidos pelo usuário: nova senha (password_new) e confirmação da nova senha (password_conf). Ambas possuem o valor admin. Nesse momento, não criticaremos o fato de a senha e sua confirmação trafegarem em claro via GET.

Sabendo que existe essa funcionalidade na aplicação, um atacante pode, por meio de engenharia social, enviar para você um e-mail chamativo, conforme o exemplo apresentado na imagem adiante.

Exemplo de possível imagem.

Atenção!

A imagem apresentada de forma alguma caracteriza um ataque. Ela é autêntica e faz referência a um sorteio legítimo. No entanto, apenas para fins de exemplo, um atacante pode utilizar uma imagem legítima para fins maliciosos.

No e-mail do atacante, ao clicar na imagem, você seria redirecionado para o seguinte endereço:

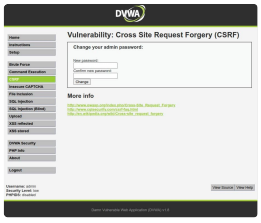
```
TERMINAL
```

Note que ele contém exatamente as URLs da funcionalidade de troca de senha apresentadas anteriormente. Assim, ao clicar no endereço do e-mail malicioso, a vítima é redirecionada para uma página de troca de senha da aplicação app (onde, supostamente, ela está autenticada). Se tudo der certo, o atacante conseguirá trocar sua senha da aplicação para 123456.

Generalizando, um ataque SSRF bem-sucedido pode forçar o usuário a realizar solicitações de alteração de estado, como transferência de fundos, alteração de endereço de e-mail e assim por diante. Se a vítima for uma conta administrativa, o CSRF (um tipo de SSRF) poderá comprometer todo o aplicativo da web.

SSRF na prática

Acesse a aplicação DVWA e selecione o desafio CSRF, como ocorre na imagem a seguir:



Desafio CSRF.

A funcionalidade apresentada realiza a troca de senha do usuário atual (no caso dessa figura, usuário admin - ver abaixo à esquerda). No exemplo apresentado, os campos foram preenchidos com 123456, enquanto a requisição foi submetida. Na URL, temos o seguinte:

```
TERMINAL
```

Podemos observar que a senha é trocada com sucesso. O problema dessa funcionalidade é que ela realiza a troca de senha sem solicitar nenhuma confirmação do usuário. Por exemplo, é uma boa prática solicitar a senha antiga antes de realizar a troca da senha. Com isso, o servidor teria alguma garantia de que a requisição está sendo realmente solicitada pelo usuário.

Ora, mas nas condições atuais, um atacante, conhecendo o problema da aplicação, poderia forjar uma URL com o seguinte conteúdo:

```
TERMINAL
```

Em seguida, ele envia essa URL para o usuário na expectativa de que o usuário esteja autenticado na aplicação DVWA (por exemplo) e que

clique na URL enviada (engenharia social). De qualquer forma, se isso ocorrer, o atacante terá conseguido trocar a senha da vítima para 123.

Imagine se esse tipo de vulnerabilidade existisse em aplicações muito utilizadas, como Facebook ou Instagram...

Prevenção

A forma mais comum de prevenção funciona incorporando dados de autenticação adicionais em solicitações, o que permite que a aplicação web detecte solicitações de locais não autorizados.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Dos motivos que levam à vulnerabilidade denominada falha de registro e monitoração de segurança, podemos citar:

A

A falta de espaço em disco para realizar o registro de eventos.

B — A não realização de backup dos registros de eventos.

C — A quantidade de aplicações web suportada pelo servidor.

D — O formato do registro de eventos, que é feito comumente no formato texto.

E — A incompatibilidade entre ferramentas, como firewall e IDS, permitindo que se faça registro de eventos de apenas um deles.

Parabéns! A alternativa B está correta.

%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3Cp%20class%3D'c-paragraph'%3EO%20registro%20de%20eventos%20e%20monitoramento%20n%C3%A3o%20s%C3%A3o%20medidas%20proativa%

Questão 2

Muitas aplicações de celular de bancos sofrem ataques similares àqueles realizados via web. Em algumas transações bancárias, o usuário, após abrir a aplicação de celular e se autenticar, ainda é solicitado, para realizar determinada transação, a informar a senha do cartão (que é diferente da senha utilizada para se autenticar na aplicação de celular). Isso ocorre, por exemplo, para transferências de grandes valores. Se não fosse isso, um atacante com acesso ao celular e ao ponto em que a aplicação do usuário já esteja aberta poderia realizar a transferência de valor para uma conta desejada. Este ataque se assemelha a qual tipo de ataque do OWASP Top 10?

A — Ao ataque de força bruta, em que é possível descobrir a senha de um usuário e utilizar a aplicação normalmente se passando pelo usuário.

B — A uma falsificação de requisição do lado do servidor. Em ambos os casos, o usuário já está autenticado na aplicação, mas o servidor precisa ter certeza de que a transação (ou requisição) é realmente um desejo do usuário.

C — A quantidade de aplicações web suportada pelo servidor.

D — O formato do registro de eventos, que é comumente texto.

E — A incompatibilidade entre ferramentas, como, por exemplo, Firewall e IDS, permitindo que se faça o registro de eventos de apenas um deles.

Parabéns! A alternativa B está correta.

%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3Cp%20class%3D'c-paragraph'%3EEm%20um%20ataque%20de%20falsifica%C3%A7%C3%A3o%20de%20requisi%C3%A7%C3%A3o%20do%20lado%2

Considerações finais

A área de segurança ligada às vulnerabilidades web é ampla, mas o conhecimento dela é essencial para um profissional desenvolvedor de sistemas ou da área de segurança da informação. Somente com o correto conhecimento de como um ataque pode acontecer e como é possível explorar determinadas vulnerabilidades, esse profissional será capaz de implementar medidas para proteger o sistema ou para mitigar os efeitos.

Nesse sentido, como verificamos neste conteúdo, o estudo do OWASP Top Ten Vulnerabilidades 2021 confirma a demanda por uma forma estruturada de apresentação das principais vulnerabilidades, destacando inclusive a importância de cada uma em ordem de ocorrência.

Nesteb bate-papo, abordamos as vulnerabilidades mais comuns nas aplicações web, trazendo a importância de tratá-las, os impactos que podem ser causados e como você deve se preparar para atuar na área de segurança.

Para ouvir o *áudio*, acesse a versão online deste conteúdo.

Explore +

Pesquise na internet sobre a ferramenta SQLMap. Ela automatiza o processo de exploração de injeção de SQL e consegue, por exemplo, em questão de minutos, obter todo o conteúdo de uma base de dados a partir de um único campo não validade.

Observe como se implementa um registro de eventos (logs) de forma remota no Linux utilizando o Rsyslog (ou seja, esse registro é realizado tanto no próprio servidor quanto em um servidor remoto).

Referências

BRASIL. **Lei nº 13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais (LGPD)**. (Redação dada pela Lei nº 13.853, de 2019). Brasília, 2018.

COMMON VULNERABILITIES AND EXPOSURES (CVE). **CVE**. Consultado na internet em: 4 jan. 2022.

CYBERHOOT. **Rainbow tables**. Publicado em: 14 maio 2020.

HOFFMAN, A. **Web application security: exploitation and countermeasures for modern web applications**. Sebastopol: O'Reilly Media, 2020.

MACÊDO, D. **Introdução ao Burp Suite**. Diego Macêdo - um pouco sobre tudo de T.I. Publicado em: 17 ago. 2016.

NATIONAL VULNERABILITY DATABASE (NVD). **NVD**. Consultado na internet em: 4 jan. 2022.

OWASP CHEAT SHEET SERIES. **Cross site scripting prevention cheat sheet**. Consultado na internet em: 4 jan. 2022.

OWASP. **OWASP broken web applications**. Consultado na internet em: 4 jan. 2022.

SUCURI. 2019 **website threat research report**. Consultado na internet em: 4 jan. 2022.

SULLIVAN, B. **Informativos sobre segurança** - ataques de negação de serviço ao XML e defesas. Microsoft. Publicado em: 17 ago. 2015.

UTO, N; MELO, S. P. **Vulnerabilidades em aplicações web e mecanismos de proteção**. Minicursos SBSeg. 2009. p. 237-283.

UTO, N. **Teste de invasão de aplicações web**. Rio de Janeiro: RNP/ESR, 2013.