Relatório de Atividade Orientada de Ensino

1. Introdução

O alinhamento inicial para este relatório foi definido em 14 de julho. Após uma reavaliação em 6 de agosto, o foco foi direcionado para documentar os aprendizados de um curso de Pentest em andamento, otimizando os estudos.

2. Vulnerabilidades Web: SQL Injection

SQL Injection é uma vulnerabilidade de segurança da web que permite a um atacante manipular as consultas (queries) que uma aplicação envia para seu banco de dados.

Impacto Potencial

O impacto de um ataque de SQL Injection bem-sucedido é severo, podendo permitir que o invasor:

- Acesse dados sensíveis que não deveria ver.
- Modifique ou exclua dados existentes no banco de dados.
- Comprometa totalmente o servidor da aplicação ou a infraestrutura de backend.

Detecção e Exemplos

A detecção pode ser feita manualmente testando os pontos de entrada da aplicação.

Exemplo de Teste: Um método comum é inserir uma condição sempre verdadeira no campo de entrada, como `' OR 1=1`. Se uma query de login vulnerável for `SELECT * FROM users WHERE username = 'input_usuario' AND password = 'input_senha'`, um atacante poderia inserir `' OR 1=1 --` no campo do usuário. A query resultante se tornaria `SELECT * FROM users WHERE username = '' OR 1=1 --' AND password = '...'`, o que faria o banco de dados retornar todos os usuários, potencialmente autenticando o invasor.

Locais Comuns da Vulnerabilidade

Embora a maioria das falhas de SQL Injection ocorra na cláusula `WHERE` de um `SELECT`, elas podem surgir em diversas partes de diferentes tipos de queries, incluindo:

- **UPDATE**: Injeção nos valores a serem atualizados ou na cláusula `WHERE`.
- **INSERT**: Injeção dentro dos valores a serem inseridos em uma nova linha.
- **SELECT**: Injeção nos nomes de tabelas, colunas ou na cláusula `ORDER BY`.

Atenção: É preciso ter muito cuidado ao usar payloads como `OR 1=1`. Uma aplicação pode reutilizar o mesmo dado de entrada em várias queries. Se essa condição atingir um comando `UPDATE` ou `DELETE` sem uma cláusula `WHERE` específica, isso pode resultar na modificação ou exclusão acidental de todos os dados de uma tabela.

3. Gerenciamento de Sistemas e Comandos Essenciais

3.1. Ambiente Linux

Gerenciamento de Usuários

- O arquivo `/etc/passwd` armazena as informações dos usuários do sistema.

- O arquivo de texto /etc/sudoers é onde as permissões de sudo são configuradas para usuários e grupos.

Comandos Principais:

- adduser 'username': Adiciona um novo usuário de forma interativa.
- deluser 'username': Remove um usuário.
- useradd -d '/dir' -s '/shell' username: Cria um usuário de forma mais controlada, definindo seu diretório home e shell padrão. Após este comando, é necessário usar passwd username para definir a senha.
- deluser username sudo: Remove um usuário do grupo sudo, revogando suas permissões administrativas.

Gerenciamento de Rede

- **Configuração de Interfaces**: O arquivo /etc/network/interfaces contém as configurações de rede. Nele, é possível definir um endereço de IP fixo (static) ou configurar para recebê-lo automaticamente do roteador (DHCP).
- **Comandos de Rede**:
    - service networking restart ou /etc/init.d/networking restart: Reinicia o serviço de rede para aplicar novas configurações.
    - dhclient eth0: Solicita um novo endereço IP do servidor DHCP para a interface eth0.
    - route ou route -n: Exibe a tabela de roteamento do sistema, com -n mostrando o endereço do gateway em formato numérico.
    - route add default gw 'gateway_ip': Adiciona uma rota padrão (gateway).
    - route del default: Remove a rota padrão, o que pode desconectar a máquina da internet.
- **Análise de Portas com netstat**: Este comando exibe as portas abertas e conexões de rede.
    - -lt: Lista as portas TCP (-t) em modo de escuta (-l).
    - -lu: Lista as portas UDP (-u) em modo de escuta (-l).
    - -lnt: Mostra o número da porta (-n) em vez do nome do serviço.
    - -lntp: Exibe a porta (-n), o protocolo TCP (-t) e o programa (-p) que a está utilizando.

3.2. Ambiente Windows (Terminal)

Manipulação de Arquivos e Diretórios

- dir: Lista arquivos e pastas. dir /a mostra também os itens ocultos, e dir /S arquivo busca por um arquivo em todos os subdiretórios.
- type arquivo.txt: Exibe o conteúdo de um arquivo de texto.
- echo texto > arquivo.txt: Cria um arquivo contendo "texto", sobrescrevendo o conteúdo se o arquivo já existir.
- echo texto >> arquivo.txt: Adiciona "texto" ao final de um arquivo, sem sobrescrevê-lo.
- attrib +h arquivo: Torna um arquivo oculto. O inverso é attrib -h arquivo.

- move arquivo destino: Move um arquivo para um novo local.
- del arquivo: Deleta um arquivo.

Rede, Processos e Usuários

- ipconfig /all: Exibe informações detalhadas sobre a configuração de rede.
- tasklist: Lista todos os processos em execução, funcionando como um Gerenciador de Tarefas no terminal.
- taskkill -pid 1234: Encerra um processo com base em seu ID de Processo (PID).
- net user: Lista todas as contas de usuário no sistema.
- net user jota senha123 /add: Cria um novo usuário chamado "jota" com a senha "senha123".
- net user jota /delete: Remove o usuário "jota".

4. Análise de Logs com Comandos de String

Comandos de manipulação de texto são essenciais para analisar arquivos de log. A seguir, a resolução de laboratórios práticos de análise.

- **Lab 1: Contar IPs únicos em um log de acesso**.
    - **Download do Log**: wget http://www.businesscorp.com.br/access.log.
    - **Extração e Contagem**: O comando cat access.log | cut -d " " -f 1 | sort -u | wc -l executa os seguintes passos:
        - cut -d " " -f 1: Corta cada linha usando o espaço como delimitador e pega a primeira coluna (o IP).
        - sort -u: Ordena os IPs e remove as duplicatas.
        - wc -l: Conta o número de linhas restantes, resultando no total de IPs únicos.
- **Lab 2: Contar requisições por IP para identificar possíveis atacantes**.
    - **Solução**: cat access.log | cut -d " " -f 1 | sort | uniq -c | sort -un.
        - uniq -c: Agrupa as linhas idênticas (IPs) e conta suas ocorrências. Isso ajuda a determinar quais IPs fizeram mais requisições, um indicativo de um possível ataque.
        - sort -un: Ordena o resultado numericamente em ordem crescente para fácil visualização.
- **Lab 3: Isolar a primeira e a última atividade de um IP específico**.
    - **Primeira Ocorrência**: cat access.log | grep 177.138.28.7 | head -n1.
    - **Última Ocorrência**: cat access.log | grep 177.138.28.7 | tail -n1.
- **Comando sed:**
    - **Para substituir valores em cada linha do arquivo:** sed 's/antigo-valor/novo-valor/' exemplo.txt
    - **Salvar as alterações no arquivo:** sed -i 's/antigo/novo/' exemplo.txt; salvar com sed -i.bak cria um arquivo backup do original.
    - O sed suporta expressão regular

5. Análise de Pacotes a Nível de Bytes

5.1. Cabeçalho TCP

O cabeçalho TCP tem uma estrutura precisa onde cada campo tem um propósito.

- **Data Offset**: Um campo de 4 bits que indica o tamanho total do cabeçalho TCP em palavras de 32 bits.
- **Flags TCP**: As flags (como SYN, ACK, FIN) são representadas por bits individuais. O valor combinado indica quais flags estão ativas. Um valor de 0x18 (24 em decimal) significa que as flags ACK (valor 16) e PSH (valor 8) estão ativadas, pois 16+8=24.

5.2. Cabeçalho Ethernet e IP

- **Quadro Ethernet**: Inicia com 6 bytes para o MAC de destino, 6 bytes para o MAC de origem e 2 bytes para o tipo de protocolo. Um valor de `08 00` indica que o protocolo encapsulado é o IP.
- **Cabeçalho IP**: O campo IHL (Internet Header Length) indica o tamanho do cabeçalho. Como cada linha do cabeçalho IP tem 4 bytes, um IHL com valor 5 significa que o cabeçalho tem 20 bytes (5 linhas x 4 bytes/linha).

6. Análise de Tráfego com TCPDump

TCPDump é um analisador de pacotes de linha de comando, servindo como uma alternativa poderosa ao Wireshark, especialmente em ambientes sem interface gráfica.

Vantagens e Características

- **Eficiência**: Roda diretamente no kernel, consumindo poucos recursos, o que o torna ideal para servidores e conexões remotas.
- **Biblioteca Comum**: Utiliza a `libpcap`, a mesma biblioteca de captura de pacotes que o Wireshark.
- **Filtros Poderosos**: Emprega a sintaxe BPF (Berkeley Packet Filter) para filtros de captura altamente específicos.

Comandos e Interpretação

- Exemplo de Captura: `tcpdump -v -i eth0 icmp -w file.pcap`
  Este comando captura (`-i eth0`) o tráfego do protocolo icmp de forma detalhada (`-v`) e salva (`-w`) em um arquivo chamado file.pcap.
- **Leitura e Detalhes**: Para ler um arquivo, usa-se a flag `-r`. A flag `-n` desativa a resolução de DNS (mostrando apenas IPs), e `-e` adiciona mais detalhes da camada de enlace, como endereços MAC.
- **Flags TCP na Saída**: No resultado do tcpdump, as flags TCP são representadas de forma abreviada: `[S]` (SYN), `[S.]` (SYN-ACK), `[.]` (ACK), `[P]` (PSH) e `[F]` (FIN).
- **Filtragem por Host**: Para filtrar pacotes de uma origem específica ao ler um arquivo: `tcpdump -vnr file.pcap src host 192.0.0.1`.

7. Bash Scripting

Scripts em Bash são uma ferramenta poderosa para automatizar tarefas no Linux.

Estrutura Básica e Execução

- **Shebang**: Todo script Bash deve começar com `#!/bin/bash`. Essa notação, conhecida como "Shebang", informa ao sistema operacional que o arquivo deve ser executado usando o interpretador Bash.
- **Execução de Comandos**: É possível executar qualquer comando do terminal diretamente dentro do script. Por exemplo, para criar um script que escaneia uma rede com o Nmap, basta adicionar a linha `nmap 192.168.1.1/24` no ponto desejado do script.

Variáveis e Entradas

- **Declaração**: Para criar uma variável, a sintaxe é `variavel=valor` (sem espaços ao redor do `=`).
- **Uso**: Para usar o valor de uma variável, utilize o cifrão ($): `echo $variavel`.
- **Entrada do Usuário**: O comando `read nome_variavel` pausa o script e armazena o que o usuário digitar na variável especificada.
- **Argumentos de Linha de Comando**: É possível passar argumentos ao executar um script (`./script.sh arg1 arg2`). Dentro do script, esses argumentos são acessados por `$1`, `$2`, e assim por diante. `echo "$1"` irá imprimir o primeiro argumento passado.

Estruturas Condicionais

Verificações:

Operador	Significado	Descrição
`-eq`	Igual a	(Do inglês: **eq**ual)
`-ne`	Diferente de	(Do inglês: **n**ot **e**qual)
`-gt`	Maior que	(Do inglês: **g**reater **t**han)
`-lt`	Menor que	(Do inglês: **l**ess **t**han)
`-ge`	Maior ou igual a	(Do inglês: **g**reater than or **e**qual)
`-le`	Menor ou igual a	(Do inglês: **l**ess than or **e**qual)

Por exemplo:

```
#!/bin/bash

# Define dois números para comparação
a=10
b=20

echo "Comparando os números: a=$a e b=$b"
echo "-----------------------------------"

# Verificando se são iguais
if [ "$a" -eq "$b" ]; then
  echo "a é igual a b."
```

```
fi

# Verificando se são diferentes
if [ "$a" -ne "$b" ]; then
  echo "a é diferente de b."
fi

# Verificando se 'a' é maior que 'b'
if [ "$a" -gt "$b" ]; then
  echo "a é maior que b."
fi

# Verificando se 'a' é menor que 'b'
if [ "$a" -lt "$b" ]; then
  echo "a é menor que b."
fi

# Verificando se 'a' é maior ou igual a 'b'
if [ "$a" -ge "$b" ]; then
  echo "a é maior ou igual a b."
fi

# Alterando 'a' para ser igual a 'b' para o próximo teste
a=20
echo ""
echo "Agora, com a=$a e b=$b"
echo "-----------------------------------"

# Verificando se 'a' é menor ou igual a 'b'
if [ "$a" -le "$b" ]; then
  echo "a é menor ou igual a b."
fi
```

Pontos Importantes:
Espaçamento é Crucial: Sempre deve haver espaços ao redor dos colchetes [ ] e do operador. ["$a" -eq "$b"] está correto. ["$a"-eq"$b"] causará um erro.

Números vs. Texto (Strings): Para comparar texto, use os operadores == (igual) e != (diferente).

Exemplo: if ["$texto1" == "$texto2"]

Alternativa Moderna: Para operações matemáticas e comparações, pode-se usar parênteses duplos ((...)), que permitem uma sintaxe mais familiar:

Exemplo: if ((a > b)); then ...

- **If/Else**: Usado para executar blocos de código com base em uma condição.
```
if [ "$variavel" == "valor" ]
then
    echo "A condição é verdadeira."
elif [ "$outra_variavel" -gt 10 ]
then
    echo "A segunda condição é verdadeira."
else
    echo "Nenhuma condição é verdadeira."
fi
```
- **Case**: Útil para comparar uma variável com múltiplos valores, funcionando como um "switch". O ;; age como um "break", finalizando a verificação daquele caso.
```
case $opcao in
    "1")
        echo "Você escolheu a opção 1."
        ;;
    "2")
        echo "Você escolheu a opção 2."
        ;;
    *)
        echo "Opção inválida."
        ;;
esac
```

Estruturas de Repetição (Loops)

- **Criação de Sequências**: É possível gerar sequências de forma simples com `echo {1..10}` (gera números de 1 a 10) ou `echo {a..z}` (gera o alfabeto). O comando `seq 1 10` tem efeito similar.
- **For Loop**: Permite iterar sobre uma lista de itens.
    - **Iterando sobre uma Sequência**:
    ```
    # Executa o nmap para IPs de 192.168.0.1 a 192.168.0.10
    for ip_final in {1..10}; do
        nmap 192.168.0.$ip_final
    done
    ```
    - **Iterando sobre um Arquivo**:
    ```
    # Executa o nmap para cada IP listado no arquivo 'ipList.txt'
    for ip in $(cat ipList.txt); do
        nmap $ip
    done
    ```

8. Vulnerabilidades Adicionais em Aplicações Web

Esta seção detalha outras vulnerabilidades críticas que foram objeto de estudo, expandindo o escopo para além do SQL Injection tradicional e abordando falhas em frameworks modernos, lógicas de manipulação de arquivos e bancos de dados não relacionais.

8.1. Prototype Pollution

Prototype Pollution é uma vulnerabilidade específica de JavaScript que ocorre quando um atacante consegue modificar o Object.prototype (o protótipo base de todos os objetos). Uma vez que o protótipo é modificado, todo objeto criado na aplicação herda as propriedades maliciosas injetadas, o que pode levar a comportamentos inesperados e brechas de segurança.

Impacto Potencial O impacto da Prototype Pollution é vasto e depende do código da aplicação, podendo resultar em:

- **Negação de Serviço (DoS):** A modificação de propriedades essenciais pode quebrar a lógica da aplicação, causando falhas e interrupções.
- **Cross-Site Scripting (XSS):** Se uma propriedade poluída for usada para renderizar HTML sem a devida sanitização (ex: innerHTML), um atacante pode injetar scripts.
- **Bypass de Autorização e Execução de Código Remoto (RCE):** Em cenários mais graves, especialmente em aplicações Node.js no lado do servidor, a poluição de protótipos pode alterar atributos como isAdmin ou manipular caminhos de arquivos e comandos, levando à execução de código arbitrário.

Detecção e Exemplo A vulnerabilidade geralmente surge em funções que realizam a fusão (merge) recursiva de objetos, clonagem de objetos ou manipulação de propriedades com base em entradas do usuário.

- **Exemplo de Fusão Vulnerável:** Considere uma função que mescla dois objetos. Se um atacante enviar um payload JSON como {"__proto__": {"isAdmin": true}}, a lógica de fusão pode acidentalmente aplicar a propriedade isAdmin: true diretamente ao Object.prototype. A partir desse ponto, qualquer novo objeto ({}) na aplicação terá a propriedade isAdmin definida como true, potencialmente concedendo privilégios de administrador.

8.2. Cross-Site Scripting (XSS) no AngularJS 1.5.8

Versões mais antigas do framework AngularJS, como a 1.5.8, tentavam mitigar XSS através de um mecanismo de *sandbox*. O objetivo era isolar as expressões do AngularJS ({{ ... }}) do resto do ambiente JavaScript, impedindo-as de acessar objetos globais perigosos como window ou document. No entanto, essa sandbox continha falhas que permitiam a um atacante "escapar" e executar JavaScript arbitrário.

Impacto Potencial O impacto é o de um ataque XSS clássico, permitindo ao invasor:

- Roubar cookies de sessão e sequestrar a conta do usuário.
- Redirecionar o usuário para páginas de phishing.

- Capturar entradas de teclado (keylogging).
- Modificar o conteúdo da página para enganar o usuário.

Detecção e Exemplo A detecção envolve injetar expressões específicas do AngularJS em campos de entrada que são refletidos na página. Para a versão 1.5.8, um payload de escape conhecido é:

{{constructor.constructor('alert(1)')()}}

- **Análise do Payload:**
    - {{ ... }}: Inicia uma expressão AngularJS.
    - constructor: Dentro da sandbox, refere-se ao construtor de um objeto (Object).
    - constructor.constructor: Acessa o construtor do construtor, que é o Function(), um construtor de função global que a sandbox deveria bloquear.
    - ('alert(1)'): Cria uma nova função anônima com o código alert(1).
    - (): Executa a função recém-criada.

Este payload explora uma falha na lógica da sandbox para obter acesso ao construtor de funções global e executar código JavaScript arbitrário no contexto da página.

8.3. Path Traversal e Local File Inclusion (LFI)

Path Traversal (também conhecido como *Directory Traversal*) é uma técnica usada para navegar pelo sistema de arquivos de um servidor e acessar diretórios e arquivos que estão fora do diretório raiz da aplicação web. Essa técnica é frequentemente explorada através de uma vulnerabilidade de **Local File Inclusion (LFI)**, que ocorre quando uma aplicação utiliza a entrada do usuário para construir o caminho de um arquivo a ser incluído ou lido no servidor.

Impacto Potencial

- **Exposição de Dados Sensíveis:** Permite a leitura de arquivos de configuração (ex: wp-config.php), código-fonte da aplicação, ou arquivos sensíveis do sistema operacional como /etc/passwd.
- **Execução de Código Remoto (RCE):** Em cenários onde o atacante pode controlar o conteúdo de um arquivo no servidor (como um arquivo de log que registra o User-Agent do atacante) e depois incluí-lo, é possível executar código arbitrário.

Detecção e Exemplo A vulnerabilidade é comumente encontrada em parâmetros de URL que especificam uma página ou arquivo a ser carregado.

- **URL Vulnerável:** https://exemplo.com/index.php?pagina=contato.php
- **Payload de Ataque:** Um atacante pode manipular o parâmetro pagina para voltar diretórios usando ../.
    - https://exemplo.com/index.php?pagina=../../../../etc/passwd
- **Técnicas de Bypass:** Muitas vezes, as aplicações filtram ../. Os atacantes podem usar codificação de URL (%2e%2e%2f) ou outras variações (..../) para contornar esses filtros.

8.4. SQL Injection em Bancos de Dados Não Relacionais (NoSQL Injection)

Embora não utilizem a linguagem SQL, os bancos de dados NoSQL (como MongoDB, Cassandra e CouchDB) também são vulneráveis a ataques de injeção. O NoSQL Injection ocorre quando a entrada do usuário não é devidamente sanitizada e é inserida diretamente em consultas ou APIs do banco de dados, permitindo que um atacante manipule a lógica da consulta.

Impacto Potencial O impacto é muito similar ao do SQL Injection tradicional:

- Bypass de autenticação.
- Extração, modificação ou exclusão de dados em toda a coleção (equivalente a uma tabela).
- Negação de Serviço (DoS).

Detecção e Exemplo (MongoDB) A vulnerabilidade geralmente ocorre quando as consultas são construídas como objetos JSON ou BSON a partir da concatenação de strings.

- **Código Vulnerável (Node.js):**

```
db.collection('users').findOne({
    username: req.body.username,
    password: req.body.password
}, function(err, user) {
    // Lógica de login
});
```

- **Payload de Ataque:** Se um atacante fornecer um nome de usuário válido (ex: admin) e, no campo de senha, injetar um operador de consulta do MongoDB, ele pode contornar a verificação.
    - **Payload no campo senha:** {"$ne": null}
    - **Consulta Resultante:** A consulta no banco se torna db.collection('users').findOne({username: 'admin', password: {"$ne": null}})
    - **Resultado:** A consulta irá autenticar o usuário admin se sua senha não for nula, sem nunca verificar o valor real da senha.

8.5. Unrestricted File Upload

Esta vulnerabilidade ocorre quando um servidor permite que os usuários enviem arquivos sem impor restrições adequadas sobre o tipo, tamanho ou conteúdo do arquivo. Isso abre uma brecha para que um atacante envie um arquivo malicioso, como uma *web shell*, e o execute no servidor.

Impacto Potencial O impacto é crítico, podendo levar ao **comprometimento total do servidor**. Uma vez que uma *web shell* é carregada e executada, o atacante obtém **Execução de Código Remoto (RCE)**, permitindo-lhe:

- Executar comandos no sistema operacional do servidor.
- Acessar e exfiltrar todos os dados da aplicação, incluindo bancos de dados.
- Usar o servidor comprometido como um pivô para atacar a rede interna.

Detecção e Estratégias de Exploração A detecção envolve identificar um formulário de upload e testar as validações implementadas.

- **Cenário de Ataque:** O objetivo é carregar uma *reverse shell* em um formato executável pelo servidor (ex: .php, .jsp, .aspx). Durante os estudos, tentou-se ocultar um payload de reverse shell em JSP dentro de outros tipos de arquivo para contornar filtros.
- **Técnicas de Bypass de Validação:**
    1. **Filtro de Extensão (Client-Side):** Se a validação for apenas no JavaScript do navegador, ela pode ser facilmente contornada desativando o JS ou interceptando a requisição com uma ferramenta como o Burp Suite.
    2. **Filtro de Content-Type (Server-Side):** Se o servidor verifica o cabeçalho Content-Type (ex: exigindo image/jpeg), um atacante pode interceptar a requisição e alterar o Content-Type de application/x-jsp para image/jpeg, mesmo enviando o arquivo shell.jsp.
    3. **Filtro de Extensão (Server-Side):**
        - **Variações de Maiúsculas/Minúsculas:** Tentar extensões como .JSP ou .jSp.
        - **Extensões Duplas:** Enviar um arquivo como shell.jsp.jpg. Dependendo da configuração do servidor, ele pode ser interpretado como um arquivo JSP.
    4. **Ocultando Payloads em Arquivos Válidos:**
        - **SVG com Script Embarcado:** Um arquivo SVG é baseado em XML e pode conter tags <script>. Se um usuário abrir o link direto para o SVG enviado, o JavaScript será executado no navegador dele, resultando em um ataque XSS armazenado.
        - **PDF/Imagem com Shell Anexada:** É possível criar um arquivo que seja ao mesmo tempo uma imagem válida e um script PHP/JSP (ex: inserindo o código do shell nos metadados EXIF de uma imagem). Se o servidor tiver uma configuração inadequada que execute arquivos com extensão .jpg através do interpretador de script, a RCE pode ser alcançada.

Elaborado por:

Jonathas Guedes Borges