Todo este texto foi escrito por mim (Tiago a.k.a TigaxMT).

Todo ele foi transcrito das videoaulas do HackerOne.

Todos créditos vão para eles.

Introdução

Bem-Vindo

Seja bem-vindo ao Hacker 101! Eu sou o Cody Brocious e serei o vosso instrutor. Nesta aula iremos aprender:

- Como identificar, explorar e corrigir as vulnerabilidades mas comuns na Web, assim como muitos outros bugs.
- Como trabalhar com criptografia corretamente
- Como desenhar e rever aplicações no ponto de vista de segurança
- Como operar como um bug bounty hunter ou um consultor de segurança
- Muito mais …

Pode usar qualquer sistema operativo, desde que consiga correr aplicações Java.

Recomendo a aprender/rever como fazer requições Web usando uma linguagem de programação à sua escolha. No caso de usar Python o módulo “requests” é o meu favorito para fazer isso.

Ferramentas que deve ter

- Burp Proxy (free edition é suficiente)
  - Este permite-nos ver toda a comunicação HTTP(S), assim como intercetar e modificar requisições, até mesmo repetir requisições.
  - É uma ferramenta que irá usar frequentemente nos seus cursos e exames.
- Firefox
  - Ele permite que configuremos um proxy no próprio navegador, que é melhor do que ter de usar o sistema operativo para o fazer.
  - Será o seu melhor amigo quando estiver a testar aplicações isoladas.

Introdução à Segurança

Está aqui para quebrar coisas antes que outro o faça, então essas vulnerabilidades podem ser corrigidas antes que os atacantes/hackers as descubram.

Para fazer isso, precisamos de perceber como os atacantes operam, quais são os seus objetivos e como eles pensam.

Pensar como um destruidor

O primeiro ponto na mentalidade de um destruidor(atacante) é: clicar num botão é a forma mais efetiva para descobrir o que ele faz.

Se não entende o que uma aplicação faz e o porquê de o fazer, vai demorar para encontrar formas de quebrar algo.

Desequilíbrio

A diferença principal entre defender e atacar é:
Quem defende tem de descobrir todos os bugs, enquanto que os atacantes só têm de encontrar alguns.
Isto significa que os atacantes terão sempre vantagem sobre os que defendem.

A principal consequência do desequilíbrio é que nunca encontrará todos os bugs, especialmente sob pressão.

Isto significa que tem de priorizar, para certificar-se que os que escaparem, terão um impacto fraco. Fazer uma avaliação precisa de áreas de alto risco é muito importante!

Objetivos dos atacantes

Quando se avalia uma aplicação, encontre cada pedaço de funcionalidade que conseguir. Assim que tiver uma lista aproximada de todas as partes da aplicação, pensem assim: Se eu fosse um atacante, qual seria o meu objetivo?

Talvez eu queira os número de cartão de crédito de um site de e-commerce, talvez eu queira destruir ou falsificar dados num sistema de monitorização.

Prioridades

Assim que tiver uma boa imagem do que o atacante quer, pode começar por classificar áreas da aplicação em termos de ganhos: Se eu comprometer a área X, ela dará informação fraca ou informação de alto valor? E a área Y?

Quando possível, perguntem aos desenvolvedores: “O que o mantém acordado de noite?” - e ele vai apontar áreas para serem verificadas.

Resultados

Para o propósito deste curso, deve incluir o seguinte para cada vulnerabilidade:

- Título – Ex: “Reflected Cross-Site Scripting em perfis”
- Severidade
- Descrição – Breve descrição acerca da vulnerabilidade
- Reprodução dos passos – Breve descrição de como reproduzir o bug, de preferência com um pequeno PoC (Prova de Conceito)
- Impacto – O que pode ser feito com esta vulnerabilidade?
- O que é afetado – Geralmente uma lista de URLs afetados.

Severidade

Esta é manipulada de forma diferente em todo o lugar, mas é recomendado basear-se na dificuldade que foi explorar a falha e também o impacto potencial que pode ter no negócio. As classificações seguintes são as que uso:

- Informativa – Falha que não tem um impacto real
- Fraca – O impacto no negócio é mínimo
- Médio – Pode afetar os utilizadores, mas sem revelar dados
- Alta – Risco potencial de revelar dados ou ajuda a explorar outras vulnerabilidades
- Crítico – Alto risco de dados pessoais serem expostos, sistemas principais comprometidos e outros impactos graves no negócio.

Exemplo

Vamos ver este exemplo comum:

```php
<?php
if(isset($_GET['name'])) {
    echo "<h1>Hello {$_GET['name']}!</h1>";
}
?>
<form method="GET">
Enter your name: <input type="input" name="name"><br>
<input type="submit">
```

Encontrar o bug

O código é bastante direto:

1. Um “name” foi passado como parâmetro via GET?
    1. Se sim, printa “Hello <name>!” numa <h1> tag
2. Printa um formulário para o utilizador introduzir o seu nome

Que código básico, o que poderá dar de errado?

Problema

O que aconteceria se fosse para esta página?

http://vulnerable.example.com/page.php?name=%3Cscript%3Ealert(1);%3C/script

O HTML seria: <h1> Hello <script>alert(1);</script>!</h1>

Reflected XSS

O que foi visto no exemplo anterior foi um reflected cross-site scripting (a.k.a Reflected XSS ou rXSS).

Basicamente um parâmetro que o atacante controla é diretamente refletido para o utilizador. Isto pode permitir injeção de HTML ou Javscript cru (dependendo onde o XSS está) e pode permitir ao atacante fazer ações no contexto de outro utilizador.

Este é um exemplo simulado, nós iremos cobrir muito mais em termos de XSS nas próprias sessões. Mas pense em todos os lugares onde o seu input é refletido no site.

Quantos desses inputs estão vulneráveis? Quantos serão seguramente validados? Ficaria surpreendido.

Próxima sessão

A próxima sessão entraremos a fundo em como os navegadores e a Web funciona em geral. Também irá aprender acerca de Cross Site Request Forgery, uma dss mais comuns e importantes vulnerabilidades.

Por agora, prepara o seu proxy and brinca com ele – veja o fluxo dos dados quando navega em vários sites.