PHDays的IDS Bypass竞赛：writeup和解决方案

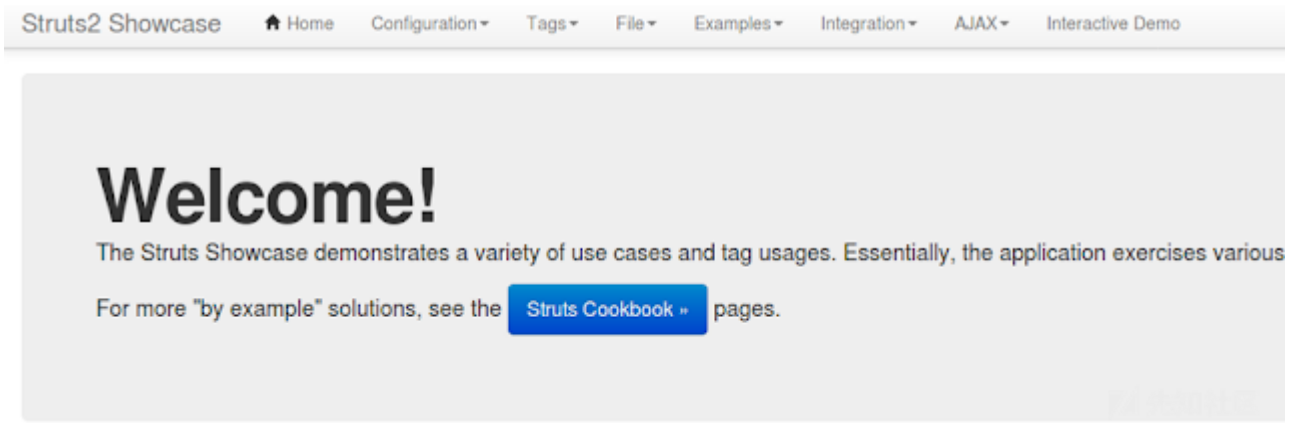
Pinging / 2019-07-09 09:10:00 / 浏览数 5159 安全技术 CTF 顶(0) 踩(0)

2019年的Positive Hack Days包括有史以来第一次IDS Bypass比赛。
参与者必须研究五个主机的网段，然后利用服务漏洞或满足特定标准（例如发送某个HTTP响应）以获得flag。
找到漏洞很容易，但是IDS会让参与者和主机之间的连接变得复杂，检查每个网络数据包。 当签名阻止连接时，通过屏幕通知参与者。
以下是有关任务的详细信息以及解决这些问题的方法。

100.64.0.11 – Struts

相比较来说，多个参与者解决了Struts任务。 使用Nmap进行端口扫描后，用户可以在端口8080上找到Apache Struts。

```
# nmap -Pn -sV -p1-10000 100.64.0.11
631/tcp  open  ipp     CUPS 2.1
8005/tcp open  mxi?
8009/tcp open  ajp13   Apache Jserv (Protocol v1.3)
8080/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
```



使用2017年的Apache Struts漏洞，攻击者可以执行OGNL注入以获取远程代码执行（RCE）。 可以使用漏洞（例如在GitHub上），但IDS可以轻松检测到它：

```
[Drop] [**] [1:1001:1] Apache Struts2 OGNL inj in header (CVE-2017-5638) [**]
```

参与者无法使用签名代码。 但是日志消息清楚地说明了它是如何工作的。 在这种情况下，签名检测到OGNL注入HTTP：

```
GET /showcase.action HTTP/1.1
Accept-Encoding: identity
Host: 100.64.0.11:8080
Content-Type: %{(#_='multipart/form-data')...
```

研究IDS的行为后，很明显IDS正在对Content-Type标头开头的组合■{做出反应。 以下有几种方法：

1 @empty_jack尝试使用他自己的字典分隔■{符号进行模糊测试，使用字符串Content-Type:■$ {进行解决。

2 模糊HTTP请求本身。@ c00lhax0r发现标题开头的空符号也会滑过IDS:Content-Type:\ 0 $ {。

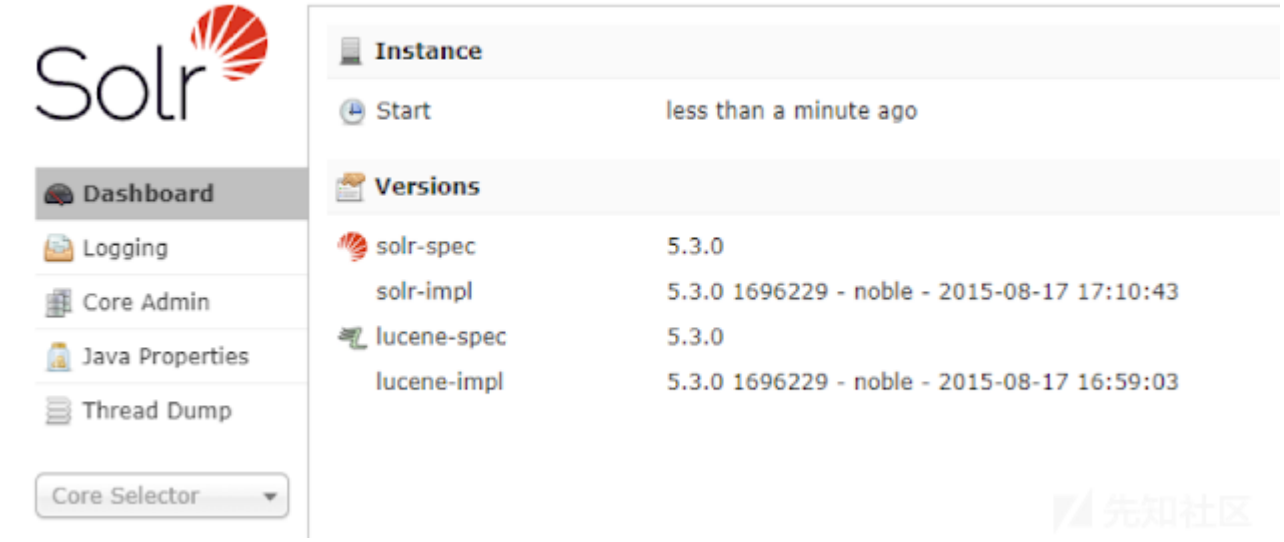
3 CVE-2017-5638的大多数漏洞都以百分号注入。 然而，一些研究此类Apache Struts漏洞的研究人员声称，注入可以很容易地从$开始。 因此组合${将绕过IDS签名并在系统上执行代码。 这是我们最初想到的解决方案。

这项任务最简单：八位参与者找到了解决方案。

100.64.0.10 — Solr

端口8983托管了一个Apache Solr服务器（用Java编写）。

```
$ nmap -Pn -sV -p1-10000 100.64.0.10
22/tcp   open  ssh     (protocol 2.0)
8983/tcp open  http    Jetty
```



查找Apache Solr 5.3.0的漏洞利用很简单：CVE-2019-0192。 攻击者可以欺骗集合中的RMI服务器的地址。
利用需要ysoserial框架，它生成Java对象链并以各种方式提供它们。 例如使用JRMP服务器。

当然，继续使用漏洞而不首先使用它只会触发IDS：

```
[Drop] [**] [1:10002700:3001] ATTACK [PTsecurity] Java Object Deserialization RCE POP Chain (ysoserial Jdk7u21) [**]
```

Jdk7u21只是30种可能的payload之一。 payload的选择取决于易受攻击的服务中使用的库。Jdk7u21小工具链仅使用Java Development Kit■JDK■版本7u21中的标准类，而CommonsCollections1链包含来自广泛使用的Apache Commons Collections 3.1的类。

攻击者可以使用其他服务器替换Solr集合中的RMI服务器地址，然后启动JRMP服务器。 Solr从攻击者指示的地址请求对象并接收恶意Java对象。
对象反序列化后，其代码在服务器上执行。

签名由序列化Java对象中的类序列触发。 从攻击者的计算机发送，以下是对象在流量中的启动方式：

```
0000  ac ed 00 05 73 72 00 17  6a 61 76 61 2e 75 74 69   ....sr.. java.uti
0010  6c 2e 4c 69 6e 6b 65 64  48 61 73 68 53 65 74 d8   l.Linked HashSet.
0020  6c d7 5a 95 dd 2a 1e 02  00 00 78 72 00 11 6a 61   l.Z..*.. ..xr..ja
0030  76 61 2e 75 74 69 6c 2e  48 61 73 68 53 65 74 ba   va.util. HashSet.
0040  44 85 95 96 b8 b7 34 03  00 00 78 70 77 0c 00 00   D.....4. ..xpw...
0050  00 10 3f 40 00 00 00 00  00 02 73 72 00 3a 63 6f   ..?@.... ..sr.:co
0060  6d 2e 73 75 6e 2e 6f 72  67 2e 61 70 61 63 68 65   m.sun.or g.apache
0070  2e 78 61 6c 61 6e 2e 69  6e 74 65 72 6e 61 6c 2e   .xalan.i nternal.
0080  78 73 6c 74 63 2e 74 72  61 78 2e 54 65 6d 70 6c   xsltc.tr ax.Templ
0090  61 74 65 73 49 6d 70 6c  09 57 4f c1 6e ac ab 33   atesImpl .WO.n..3
```

这个任务的解决方案很简单。 签名明确命名为Jdk7u21。 要绕过签名必须尝试其他工具链。例如CommonsCollections， IDS没有其他连锁的签名。
然后参与者将获得系统上的shell并读取标志。 五名参与者成功完成了这项任务。

100.64.0.12 – SAMR

这是最棘手和最有趣的任务之一。 目标是具有开放端口445的Windows计算机。该标志分为两个用户名，因此完成任务需要枚举所有Windows用户。

当然，MS17-010和其他漏洞在这台计算机上不起作用。 可以使用脚本获取用户列表，例如来自Nmap或Impacket的脚本：

```
$ python samrdump.py 100.64.0.12
Impacket v0.9.15 - Copyright 2002-2016 Core Security Technologies

[*] Retrieving endpoint list from 100.64.0.12
[*] Trying protocol 445/SMB…
Found domain(s):
. SAMR
. Builtin
[*] Looking up users in domain SAMR
[-] The NETBIOS connection with the remote host timed out.
[*] No entries received.
```

两个脚本都在端口445上向计算机发送DCERPC请求。但事情并非如此简单。 一些数据包被IDS阻止，不仅触发一个，而且触发两个签名：

```
[**] [1:2001:2] SAMR DCERPC Bind [**]
[Drop] [**] [1:2002:2] SAMR EnumDomainUsers Request [**]
```

第一个签名检测到与SAMR的连接并标记TCP连接。 第二个签名由SAMR EnumDomainUsers请求触发。
SAMR提供了获取用户列表的其他方法：QueryDisplayInfo，QueryDisplayInfo2和QueryDisplayInfo3。 所有这些都被签名阻止了。

DCERPC协议和Windows服务包含大量远程管理功能。 大多数着名的工具，如PsExec和BloodHound，都使用DCERPC。
SAMR（“SAM远程协议”）允许使用主机上的帐户，包括用户列表的枚举。

要发出EnumDomainUsers请求，这是Impacket的作用：

```
DCERPC     330 Bind: call_id: 2, Fragment: Single, 3 context items: SAMR V1.0 (32bit NDR),
SMB2       138 Write Response
SMB2       171 Read Request Len:1024 Off:0 File: samr
DCERPC     254 Bind_ack: call_id: 2, Fragment: Single, max_xmit: 4280 max_recv: 4280, 3 re
SAMR       306 Connect5 request
SAMR       234 Connect5 response
SAMR       230 EnumDomains request
SAMR       378 EnumDomains response
SAMR       286 LookupDomain request,
SAMR       238 LookupDomain response
SAMR       258 OpenDomain request
SAMR       218 OpenDomain response
SAMR       234 EnumDomainUsers request
```

通过SMB建立到SAMR的DCERPC连接，并且在SAMR上下文中发送所有后续请求。 签名由屏幕截图中的第一个和最后一个数据包触发。

在比赛中，为此任务提供了两条线索：

> 尝试使IDS生成2个警报。 仔细看第一个。
>
> 我们知道该协议的哪些连接命令？

我们的想法是开始考虑DCERPC和不同的连接方法。 在用于连接和更改上下文的可用PDU列表中，我们找到了Bind和Alter Context命令。 Alter Context允许在不中断连接的情况下更改当前上下文。

要获得解决方案，我们需要重新编写samrdump脚本：

1 绑定到其他服务，例如UUID 3919286a-b10c-11d0-9ba8-00c04fd92ef5。
2 使用Alter Context切换到SAMR。
3 发出EnumDomainUsers请求。

所有改变均满足以下条件：

```
<         dce.bind(samr.MSRPC_UUID_SAMR)
---
>         dce.bind(uuid.uuidtup_to_bin(("3919286a-b10c-11d0-9ba8-00c04fd92ef5", "0.0")))
>         dce.alter_ctx(samr.MSRPC_UUID_SAMR)
>         dce._ctx = 1
```

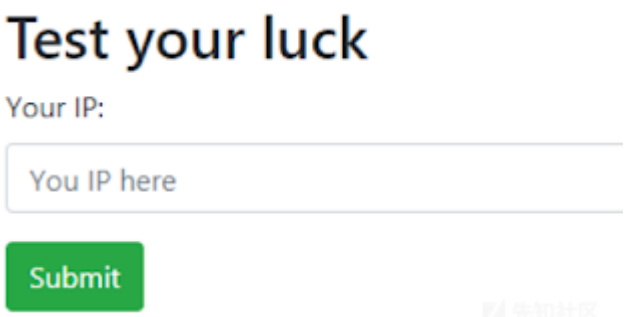
竞赛获胜者@psih1337还提出了另一种解决方案。 EnumDomainUsers返回按SID而不是按名称排序的用户列表。 但SID不是随机数。
例如，LocalSystem帐户的SID是S-1-5-18。 对于手动创建的组或用户，SID为1000或更高。

因此，如果我们手动强制使用1000到2000之间的SID，很可能会找到正在寻找的帐户。 在我们的例子中，SID分别为1008和1009。

此任务需要了解DCERPC协议以及调查Windows基础结构的一些经验。 @ psih1337是唯一一个解决这个任务的人。

100.64.0.13 – DNSCAT

端口80托管一个网页，其中包含用于输入IP地址的表单。



如果键入自己的IP地址，端口53将接收UDP，如下所示：

```
17:40:45.501553 IP 100.64.0.13.38730 > 100.64.0.187: 61936+ CNAME? dnscat.d2bc039ce800000000d6eae8eae3bf81fd84d1695f5888aba8dc
17:40:45.501639 IP 100.64.0.187 > 100.64.0.13: ICMP 100.64.0.187 udp port domain unreachable, length 209
17:40:46.520457 IP 100.64.0.13.38730 > 100.64.0.187: 21842+ TXT? dnscat.7f4e039ce800000000d6eae8eae3bf81fd84d1695f5888aba8dcec
17:40:46.520546 IP 100.64.0.187 > 100.64.0.13: ICMP 100.64.0.187 udp port domain unreachable, length 209
```

它显然是DNSCAT，一种用于DNS隧道的工具。 在表单中键入IP地址时，DNSCAT客户端会尝试连接到该地址。
如果尝试成功，则服务器（换句话说，参与者）在竞赛计算机上获得shell并收集标志。

当然，如果我们只是尝试提升DNSCAT服务器并接受连接，那就没有这样的运气了：

```
[Drop] [**] [1:4001:1] 'dnscat' string found in DNS response [**]
```

IDS签名由来自我们服务器的流量中的字符串“dnscat”触发 - 从消息中可以清楚地看出这一点。 模糊或加密流量将无法正常工作。

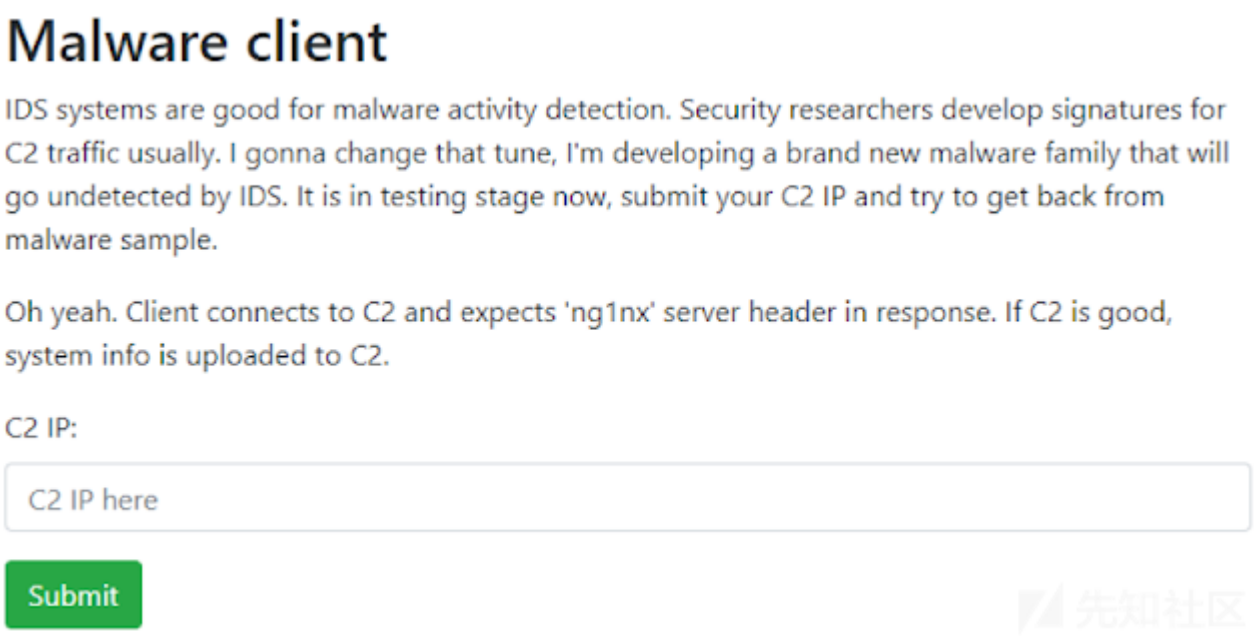
但是看一下客户端代码，我们发现检查不是太严格。 响应根本不需要包含“dnscat”字符串！
我们只需要从代码中删除字符串，或者在NetSED的帮助下即时替换它。 在运行中交换它要容易得多，但这里是服务器代码的补丁，以防万一：

```
diff -r dnscat2/server/libs/dnser.rb dnscat2_bypass/server/libs/dnser.rb
<           segments << unpack("a#{len}")
>           segments << [unpack("a#{len}")[0].upcase]

<         name.split(/\./).each do |segment|
>         name.upcase.split(/\./).each do |segment|

diff -r dnscat2/server/tunnel_drivers/driver_dns.rb dnscat2_bypass/server/tunnel_drivers/driver_dns.rb
<         response = (response == "" ? "dnscat" : ("dnscat." + response))
>         response = (response == "" ? "dnsCat" : ("dnsCat." + response))
```

五位参与者迎接了挑战。

100.64.0.14 – POST

没有参赛人员完成此任务。



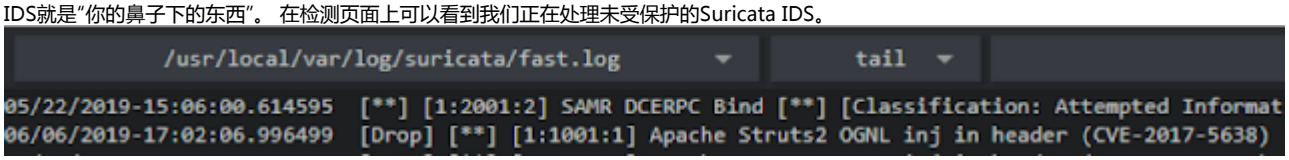
我们看到了现在熟悉的输入IP地址的形式，邀请我们参与测试新的恶意软件。 其中一个新技巧是以某种未知的方式绕过IDS。
要获取该标志，您需要做的就是发送HTTP标头“Server:ng1nx”作为响应。

正如预期的那样，我们在我们的IP地址处获得GET请求并发送响应，该响应被IDS阻止。

```
[Drop] [**] [1:5002:1] 'ng1nx' Server header found. Malware shall not pass [**]
```

以下是给与会者的提示：
有时，看起来很难的任务是最简单的。 如果没有什么东西看起来很脆弱，也许你在鼻子底下遗漏了什么东西？

IDS就是“你的鼻子下的东西”。 在检测页面上可以看到我们正在处理未受保护的Suricata IDS。



搜索“Suricata IDS Bypass”以及获得的第一个链接指向CVE-2018-6794。
如果正常的TCP握手过程中断并且数据在过程完成之前发送，则此漏洞允许绕过数据包检查。 它看起来像这样：

```
Client    ->  [SYN] [Seq=0 Ack=0]           ->  Evil Server   # 1/2
Client    <- 2="" ack="1] " eq="0" evil="" font="" nbsp="" server="">
Client    <- ack="1] " data="" eq="1" evil="" font="" here="" nbsp="" server="">
Client    <- ack="1] " eq="83" evil="" font="" nbsp="" server="">
Client    ->  [ACK] [Seq=1 Ack=84]          ->  Evil Server   # 3/2
Client    ->  [PSH, ACK] [Seq=1 Ack= 4]     ->  Evil Server
```

我们下载漏洞，将字符串更改为“ng1nx”，禁用内核重置（RST）数据包，然后运行它。

如上所述，没有人能够得到这个flag，尽管一些参与者非常接近。

结论

49人报名参加比赛，其中12人成功收集了至少一个flag。 任务可以有多个解决方案，尤其是涉及SMB和DCERPC的任务。

■■■■■■■■■■■■http://blog.ptsecurity.com/2019/07/ids-bypass-contest-at-phdays-writeup.html

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板