Java FTP CRLF Injection

zhouliu / 2017-11-16 11:04:46 / 浏览数 2513 技术文章 技术文章 顶(0) 踩(0)

文章略长，实战意义有限，慎读。

0x00 前言
前两天看了个博客http://blog.blindspotsecurity.com/ ，
故成此文，算是翻译加读后感。英文不算太烂的建议阅读原文。

0x01 概述
看了题目，熟悉python的应该会联想到其urllib库的头注入（CVE-2016-5699）。这里是Java在处理FTP流的时候存在类似的注入。利用这个缺陷可以在很多场景实施攻击（

0x02 基础知识
要明白这个姿势需要储备一定的ftp知识。ftp比较简单，以下是不负责任的要点概括：
我们知道ftp服务器默认端口是21，其实服务端在这个端口和客户端建立的连接只是用于传输命令，这个连接叫做"control connection"。而用于传输数据的端口，服务端默认监听在20端口中，这个连接叫做"data connection"。但是需要注意的是，服务端的data connection端口并不一定是20。这里引出ftp的两种工作模式：主动（active）模式和被动（passive）模式。

在主动模式中:

- 客户端从任意端口n(n≥1024）与服务端21端口建立"control connection"；
- 客户端通过端口n发送PORT指令，通知服务端自己监听的"data connection"端口为n+1（默认是n+1但是不总是）；
- 服务端从20端口与客户端的n+1端口建立"data connection"。

以下是我测试访问ftp://cddis.gsfc.nasa.gov/pub/gps/igsmail/igsmess.6988下载文件的部分报文：

PORT指令格式为：
PORT h1,h2,h3,h4,p1,p2
1）h1-h4对应IP 4个8bit地址，如10,1,2,4表示10.1.2.4；
2）p1,p2对应端口，计算方式：port = p1 2^8 + p2 2^0

在被动模式中:

- 客户端从任意非特权端口n(n≥1024）与服务端21端口建立"controlconnection"；
- 客户端通过端口n发送PASV指令，通知服务端采用被动模式建立"data connection"；
- 服务端从任意非特权端口m (m≥1024）监听"data connection"；
- 客户端从端口n+1（默认是n+1但是不总是）与服务端m端口建立 "dataconnection"。

0x03 细节
本文讨论的场景是主动模式，观察主动模式的图示，会发现客户端会监听一个端口等待服务端回连。假设客户端前面有防火墙不允许入站流量，岂不是无法建立连接？大部分

上面所述如果清楚了，防火墙欺骗就跃然纸上了。如果攻击者能控制受害客户端主动发送PORT指令，并指定特定端口x，那么防火墙会被欺骗建立NAT规则，进而受害客户端

但是有两个棘手的问题需要解决。

(1) 受害客户端的内网IP地址
我们想控制受害客户端发送PORT命令，必须知道其内网IP（回顾PORT命令），如果IP不正确，防火墙不会设置相应的规则。怎么获取其内网IP呢？我从原文中看不是很透彻
FTP clients will attempt to initiate a passive session to retrieve the z.txt file, but if the attacker's FTP server rejects the PASV command, then the client will fall back to classic mode and send a PORT command. Since the port used for the control channel is non-standard, it is unlikely that a stateful firewall at the victim's site will attempt to interpret and translate the PORT commands on this session. That will cause the internal IP address of the victim to be leaked to the attacker.
字面意思是先尝试让客户端访问攻击者的ftp server，客户端首先会尝试以被动模式建立"data connection"，服务端拒绝其PASV命令，迫使客户端采用主动模式，当客户端采用主动模式，会发送PORT指令，指令会带上内网IP，从而造成泄露。

为了弄清楚，看了ftp相关的源码，看完就明朗多了。
1) openDataConnection

```
private Socket openDataConnection(String var1) throws FtpProtocolException, IOException {
       try {
           return this.openPassiveDataConnection(var1);
       } catch (FtpProtocolException var14) {
           String var4 = var14.getMessage();
           if (!var4.startsWith("PASV") && !var4.startsWith("EPSV")) {//■■■■■■■■■■
               throw var14;
           } else if (this.proxy != null && this.proxy.type() == Type.SOCKS) {
               throw new FtpProtocolException("Passive mode failed");
           } else {
```

```
                ServerSocket var3 = new ServerSocket(0, 1, this.server.getLocalAddress());

                Socket var2;
                try {
                    InetAddress var15 = var3.getInetAddress();//IP■■
                    if (var15.isAnyLocalAddress()) {
                        var15 = this.server.getLocalAddress();
                    }

                    String var5 = "EPRT |" + (var15 instanceof Inet6Address ? "2" : "1") + "|" + var15.getHostAddress() + "|" +
                    if (!this.issueCommand(var5) || !this.issueCommand(var1)) {
                        var5 = "PORT ";
                        byte[] var6 = var15.getAddress();

                        for(int var7 = 0; var7 < var6.length; ++var7) {
                            var5 = var5 + (var6[var7] & 255) + ",";
                        }

                        var5 = var5 + (var3.getLocalPort() >>> 8 & 255) + "," + (var3.getLocalPort() & 255);
                        this.issueCommandCheck(var5);
                        this.issueCommandCheck(var1);
                    }

                    if (this.connectTimeout >= 0) {
                        var3.setSoTimeout(this.connectTimeout);
                    } else if (defaultConnectTimeout > 0) {
                        var3.setSoTimeout(defaultConnectTimeout);
                    }

                    var2 = var3.accept();
                    if (this.readTimeout >= 0) {
                        var2.setSoTimeout(this.readTimeout);
                    } else if (defaultSoTimeout > 0) {
                        var2.setSoTimeout(defaultSoTimeout);
                    }
                } finally {
                    var3.close();
                }

                if (this.useCrypto) {
                    try {
                        var2 = this.sslFact.createSocket(var2, this.serverAddr.getHostName(), this.serverAddr.getPort(), true);
                    } catch (Exception var12) {
                        throw new IOException(var12.getLocalizedMessage());
                    }
                }

                return var2;
            }
        }
    }
```

openDataConnection打开“data connection”，调用了openPassiveDataConnection，尝试调用被动模式，先看看openPassiveDataConnection，再回头看openDataConnection。

2）openPassiveDataConnection

```
private Socket openPassiveDataConnection(String var1) throws FtpProtocolException, IOException {
        InetSocketAddress var4 = null;
        String var2;
        int var3;
        Matcher var5;
        String var6;
        if (this.issueCommand("EPSV ALL")) {
            this.issueCommandCheck("EPSV");
            var2 = this.getResponseString();
            if (epsvPat == null) {
                epsvPat = Pattern.compile("^229 .* \\(\\|\\|\\|(\\d+)\\|\\)");
            }
```

```
    var5 = epsvPat.matcher(var2);
    if (!var5.find()) {
        throw new FtpProtocolException("EPSV failed : " + var2);//■■■■■■■■■■
    }

    var6 = var5.group(1);
    var3 = Integer.parseInt(var6);
    InetAddress var7 = this.server.getInetAddress();
    if (var7 != null) {
        var4 = new InetSocketAddress(var7, var3);
    } else {
        var4 = InetSocketAddress.createUnresolved(this.serverAddr.getHostName(), var3);
    }
} else {
    this.issueCommandCheck("PASV");
    var2 = this.getResponseString();
    if (pasvPat == null) {
        pasvPat = Pattern.compile("227 .* \\(?(\\d{1,3},\\d{1,3},\\d{1,3},\\d{1,3}),(\\d{1,3}),(\\d{1,3})\\)?");
    }

    var5 = pasvPat.matcher(var2);
    if (!var5.find()) {
        throw new FtpProtocolException("PASV failed : " + var2);/■■■■■■■■■■
    }

    var3 = Integer.parseInt(var5.group(3)) + (Integer.parseInt(var5.group(2)) << 8);
    var6 = var5.group(1).replace(',', '.');
    var4 = new InetSocketAddress(var6, var3);
}

Socket var9;
if (this.proxy != null) {
    if (this.proxy.type() == Type.SOCKS) {
        var9 = (Socket)AccessController.doPrivileged(new PrivilegedAction<Socket>() {
            public Socket run() {
                return new Socket(FtpClient.this.proxy);
            }
        });
    } else {
        var9 = new Socket(Proxy.NO_PROXY);
    }
} else {
    var9 = new Socket();
}

InetAddress var10 = (InetAddress)AccessController.doPrivileged(new PrivilegedAction<InetAddress>() {
    public InetAddress run() {
        return FtpClient.this.server.getLocalAddress();
    }
});
var9.bind(new InetSocketAddress(var10, 0));
if (this.connectTimeout >= 0) {
    var9.connect(var4, this.connectTimeout);
} else if (defaultConnectTimeout > 0) {
    var9.connect(var4, defaultConnectTimeout);
} else {
    var9.connect(var4);
}

if (this.readTimeout >= 0) {
    var9.setSoTimeout(this.readTimeout);
} else if (defaultSoTimeout > 0) {
    var9.setSoTimeout(defaultSoTimeout);
}

if (this.useCrypto) {
    try {
        var9 = this.sslFact.createSocket(var9, var4.getHostName(), var4.getPort(), true);
    } catch (Exception var8) {
```

```
                throw new FtpProtocolException("Can't open secure data channel: " + var8);
            }
        }

        if (!this.issueCommand(var1)) {
            var9.close();
            if (this.getLastReplyCode() == FtpReplyCode.FILE_UNAVAILABLE) {
                throw new FileNotFoundException(var1);
            } else {
                throw new FtpProtocolException(var1 + ":" + this.getResponseString(), this.getLastReplyCode());
            }
        } else {
            return var9;
        }
    }
```

反复看openPassiveDataConnection结合原文，如果在建立被动模式时服务端返回的状态码不是229/227（见上面红色框代码片段）,则会抛出异常，异常信息EPSV.../PAV
if (!var4.startsWith("PASV") && !var4.startsWith("EPSV")) {
顺着openDataConnection方法往下看，其会拼接 EPRT | 2（1）| ip | port，这里的ip正是我们需要的（原文说PORT命令似乎并不准确）。

于是，我参照https://github.com/jacklam718/ftp/blob/master/ftpServer.py改造了一个ftp服务端，跑起来后本地wireshark抓包，从结果看符合猜想。

(2) 报文对齐
由于FTP是基于行的同步协议，意味着任何一端一次写入一行，然后等待另一端响应才能再写入一行，也就是说任何一端一次只能写入一条命令。
因此，假设我们通过以下链接注入：
ftp://u:p@evil.com/foodir%0APORT%2010,1,1,1,5,57/z.txt
客户端发送的报文大致如下：
USER u
--Packet 2--
PASS p
--Packet 3--
TYPE I
--Packet 4--
CWD foodir
PORT 10,1,1,1,5,57
--Packet 5--

想要我们注入的PORT命令有效，必须恰好在报文的起始位置。
显然，最直接的方式是我们可以将foodir换成足够长的字符串，这样发送CWD
foodir的报文正好能填满TCP报文的最大长度，从而将PORT命令"挤到"另一个报文。但是MTU比较大，因此在实际攻击中可能过于招眼而显得不实际。
换另一个思路，因为ftp服务端攻击者可以控制，所以可以将MTU设置足够小，这样只需少量填充。

0x04 PoC
在分析的时候，我做了部分实验，防火墙这一块还没有解决，一是没有现成的防火墙（应该通过软件可以模拟）；二是防火墙配置还需要进一步看。
很遗憾，我也在等作者的PoC。

0x05 参考链接
http://blog.blindspotsecurity.com/
http://www.ietf.org/rfc/rfc959.txt
http://www.enyo.de/fw/security/java-firewall/
https://github.com/jacklam718/ftp/blob/master/ftpServer.py

上一篇：某开源框架从信息泄露到后台失守 下一篇：一步一步PWN路由器之uClibc...
1. 2 条回复


wooyun 2017-11-16 16:23:10

不错

0 回复Ta


wefgod 2017-11-21 17:11:07

没搞懂，暂时因为特殊原因没法开原文。

0 回复Ta

登录 后跟帖

先知社区

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