编写加密传输爆破插件 jsEncrypter | 回忆飘如雪

我曾经听某大牛所过两句话：

```
1．我们能入侵最先进的系统，却不能阻止用户使用弱口令。

2．当一个系统的用户超过1000+,那么弱口令一定存在！
```

不管这两句话是否属实，但都说明了一个问题，弱口令虽然简单，但是很难完全消除。因为它的问题不是出现在技术层面，而是在人性！所以每次渗透测试我都比较注重弱口令的检测。

当一个系统没有对登录次数进行限制时，我们就可以考虑进行爆破了。在我经验中，爆破遇到了以下三个难点：

序号	情况	解决
1	验证码	验证码有的可以绕过， 无法绕开也已经存在识别验证码的插件。
2	token	token 问题，使用 burp Suite 完全可以解决。
3	加密传输	目前解决方案比较少，对应的工具基本没有找到。

今天特地对第三种情况进行解决，所以有了此文！

针对加密传输问题，freeBuf 上的《对登录中账号密码进行加密之后再传输的爆破的思路和方式》写的挺好，作者提供了 4 种思路去解决，比我思考的全面。我最初的解决方案类似文章中的第四种思路，今天的解决方案是写一个 Burp 插件，和文章中的第一种思路类似但又有点区别。

0x01 流程

上一个流程图，给大家捋一捋插件运行的整个流程。

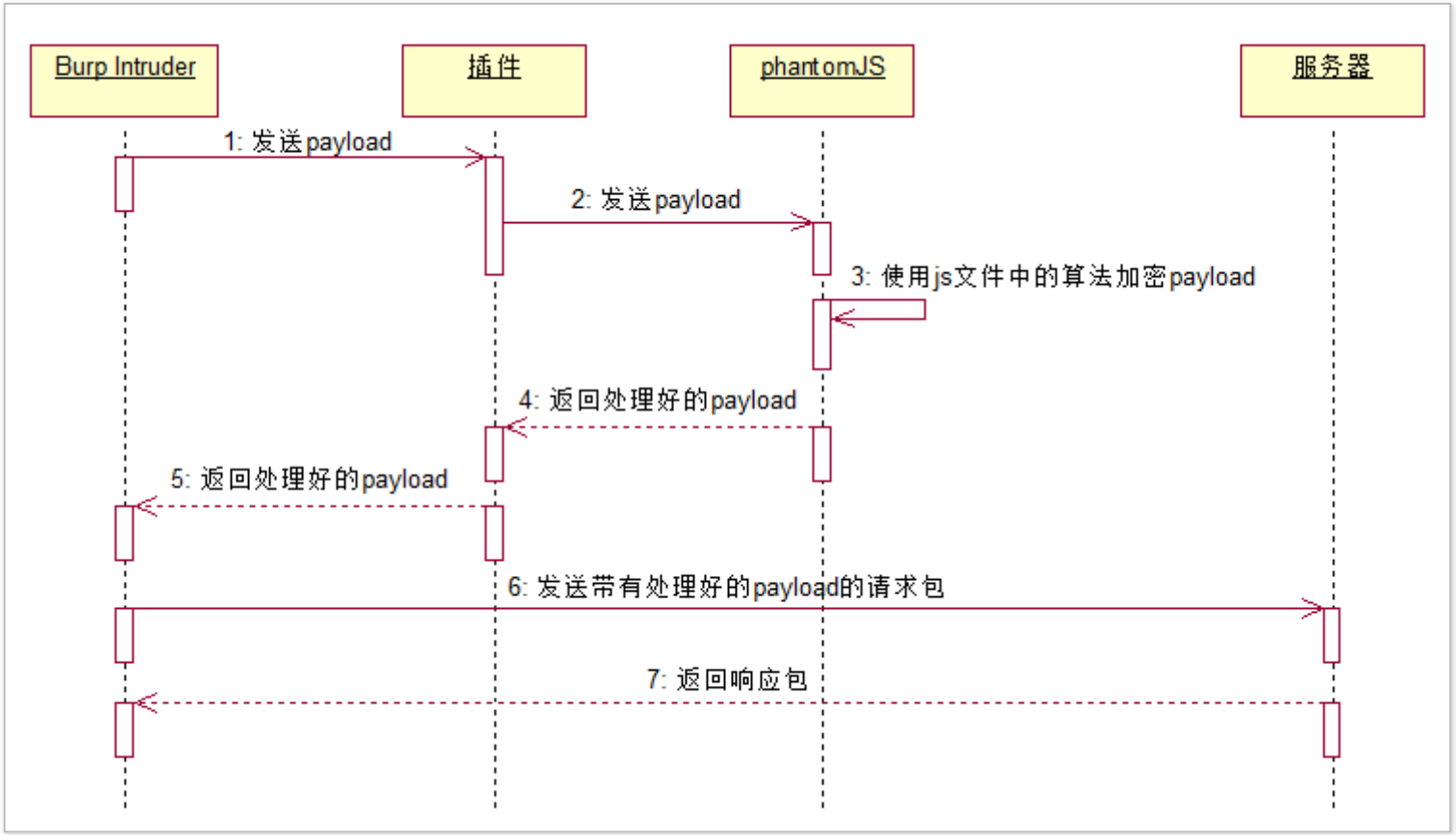

图 1 - 流程图

0x02 开发

插件核心代码

我们的插件实现对 payload 的处理，所以一定要实现 Burp Suite APIs 的 接口的 方法

`IIntruderPayloadProcessor` 接口的 `processPayload` 方法。

```
public byte[] processPayload(byte[] currentPayload, byte[] originalPayload, byte[]
            byte[] newpayload ="".getBytes();
            String payload = new String(currentPayload); //获取当前paylaod
            CloseableHttpClient client = HttpClients.createDefault(); //新建一个Ht
            HttpPost httpPost = new HttpPost(gui.getURL()); //新建一个post请求
            try {
                    List nameValuePairs = new ArrayList(1);
                    nameValuePairs.add(new BasicNameValuePair("payload",payload));
                    httpPost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
                    CloseableHttpResponse response = client.execute(httpPost); //
                    //获取phantomJS处理好的结果
                    String responseAsString = EntityUtils.toString(response.getEnt
                    newpayload = helpers.stringToBytes(responseAsString);

            } catch (Exception e) {
                    stderr.println(e.getMessage());
```

```
17                      newpayload = "JsEncrypter cannot connect phantomJS!".getBytes(
18                  }
19                  return newpayload; //返回处理好的payload给Burp Suite
20          }
```

phantomJS 脚本编写

phantomJS 是一个没有界面的浏览器，除了不能浏览，其他的和正常浏览器一样。使用它来执行我们编写好的脚本。

phantomJS 下载地址：`http://phantomjs.org/download.html`

由于每个网站前端加密传输的算法一样，所以每次引入的 js 都不同，调用加密函数的代码也不仅相同。鉴于以上情况，为了每次不用重复写一些固定的代码，我们写一个模板代码。每次使用时，只要填写好引入 js 的文件名，以及实现好在 `js_encrypt()` 函数体调用加密算法对 payload 进行加密处理即可

数据调用加密算法对 payload 进行加密处理即可。

phatomJS 脚本模板代码

```
/**
 * author: c0ny1
 * date: 2017-12-16
 */

var webserver = require('webserver');
server = webserver.create();

var host = '127.0.0.1';
var port = '1664';

// 加载实现加密算法的js脚本
var wasSuccessful = phantom.injectJs('xxx.js');/*引入实现加密的js文件*/

```

```
15  // 处理函数
16  function js_encrypt(payload){
17          var newpayload;
18          /**********在这里编写调用加密函数进行加密的代码***********/
19
20          /*********************************************************/
21          return newpayload;
22  }
23
24  if(wasSuccessful){
25          console.log("[*] load js successful");
26          console.log("[!] ^_^");
27          console.log("[*] jsEncrypterJS start!");
28          console.log("[+] address: http://"+host+":"+port);
29  }else{
30          console.log('[*] load js fail!');
31  }
32
```

```
var service = server.listen(host+':'+port,function(request, response){
        if(request.method == 'POST'){
                var payload = request.post['payload'];
         var encrypt_payload = js_encrypt(payload);
         console.log('[+] ' + payload + ':' + encrypt_payload); //显示原始payload和加
                response.statusCode = 200;
                response.write(encrypt_payload.toString()); //返回处理好的payload
         response.close();
        }else{
                  response.statusCode = 200;
                  response.write("^_^\n\rhello jsEncrypter!");
           response.close();
        }
});
```

0x03 演示

完整的代码请移步 github: http://github.com/c0ny1/jsEncrypter

大家自行下载，编译好, 最后加载到 Burp Suite 中!

(1) 靶机搭建

项目 jsEncrytper/server 目录下提供一个 php 编写的靶机，我们用 phpStudy 把

他运行起来。靶机目前支持的加密算法有 7 中：
图 2 - 插件界面

- base64 （PS：严格来说 base64 是一种编码，不是一种加密算法）
- md5
- sha1
- sha254
- sha384
- sha512
- RSA

我们选择 sha1 来进行演示。

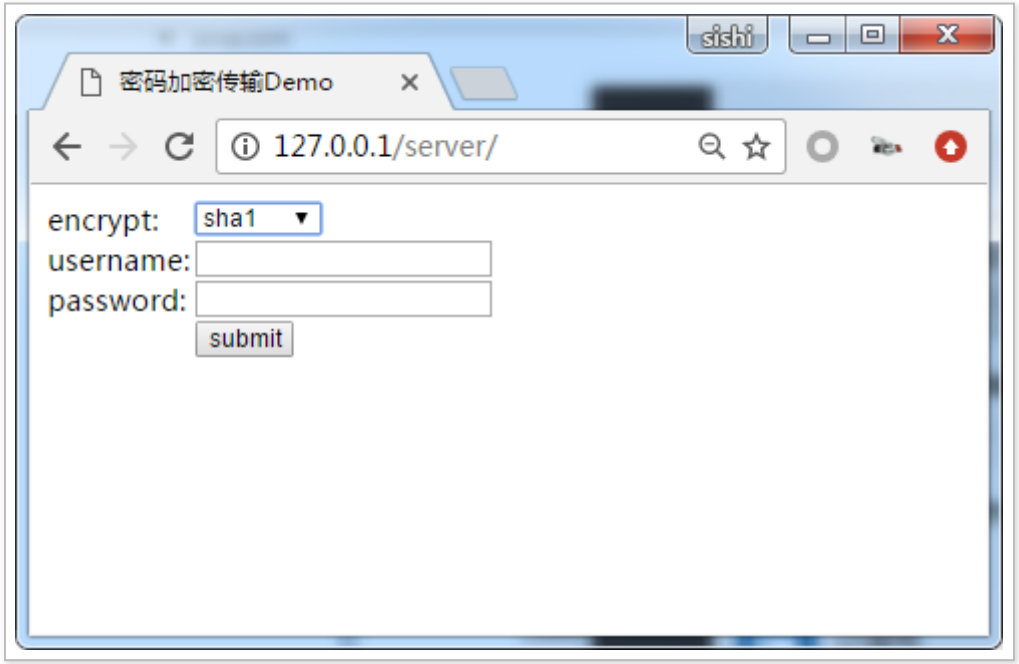

图 3 - 靶机

(2) 编写 phantomJS 脚本

- 通过查看靶机页面的 js 代码，我们知道实现 sha1 加密的是 `sha1.js` 这个文件，我们将它下载下来。
- 复制 phantomJS 模板代码 `jsEncrypter/js/jsEncrypter_base.js` 文件，改名为 jsEncrypter_sha1.js。
- 在脚本中加载 `sha1.js`，然后在 `js_encrypt` 函数中实现调用加密函数对传入的 payload 进行加密处理，即可。

完整代码如下：

```
1  var webserver = require('webserver');
2  server = webserver.create();
3
4  var host = '127.0.0.1';
5  var port = '1664';
6
7  // 加载实现加密算法的js脚本
```

```
8  var wasSuccessful = phantom.injectJs('sha1.js');/*引入实现加密的js文件*/
9
10 // 处理函数
11 function js_encrypt(payload){
12         var newpayload;
13         /**********在这里编写调用加密函数进行加密的代码************/
14         newpayload = hex_sha1(payload);
15         /*****************************************************/
16         return newpayload;
17 }
18
19 if(wasSuccessful){
20         console.log("[*] load js successful");
21         console.log("[!] ^_^");
22         console.log("[*] jsEncrypterJS start!");
23         console.log("[+] address: http://"+host+":"+port);
24 }else{
25         console.log('[*] load js fail!');
```

```
26  }
27
28  var service = server.listen(host+':'+port,function(request, response){
29          if(request.method == 'POST'){
30                  var payload = request.post['payload'];
31           var encrypt_payload = js_encrypt(payload);
32           console.log('[+] ' + payload + ':' + encrypt_payload);
33                  response.statusCode = 200;
34                  response.write(encrypt_payload.toString());
35           response.close();
36          }else{
37                    response.statusCode = 200;
38                    response.write("^_^\n\rhello jsEncrypt!");
39             response.close();
40          }
41  });
```

(3) 运行 phantomJS 脚本

```
λ phantomjs.exe jsEncrypter_sha1.js
[*] load js successful
[!] ^_^
[*] jsEncrypterJS start!
[+] address: http://127.0.0.1:1664
```

(4) 测试是否能成功加密

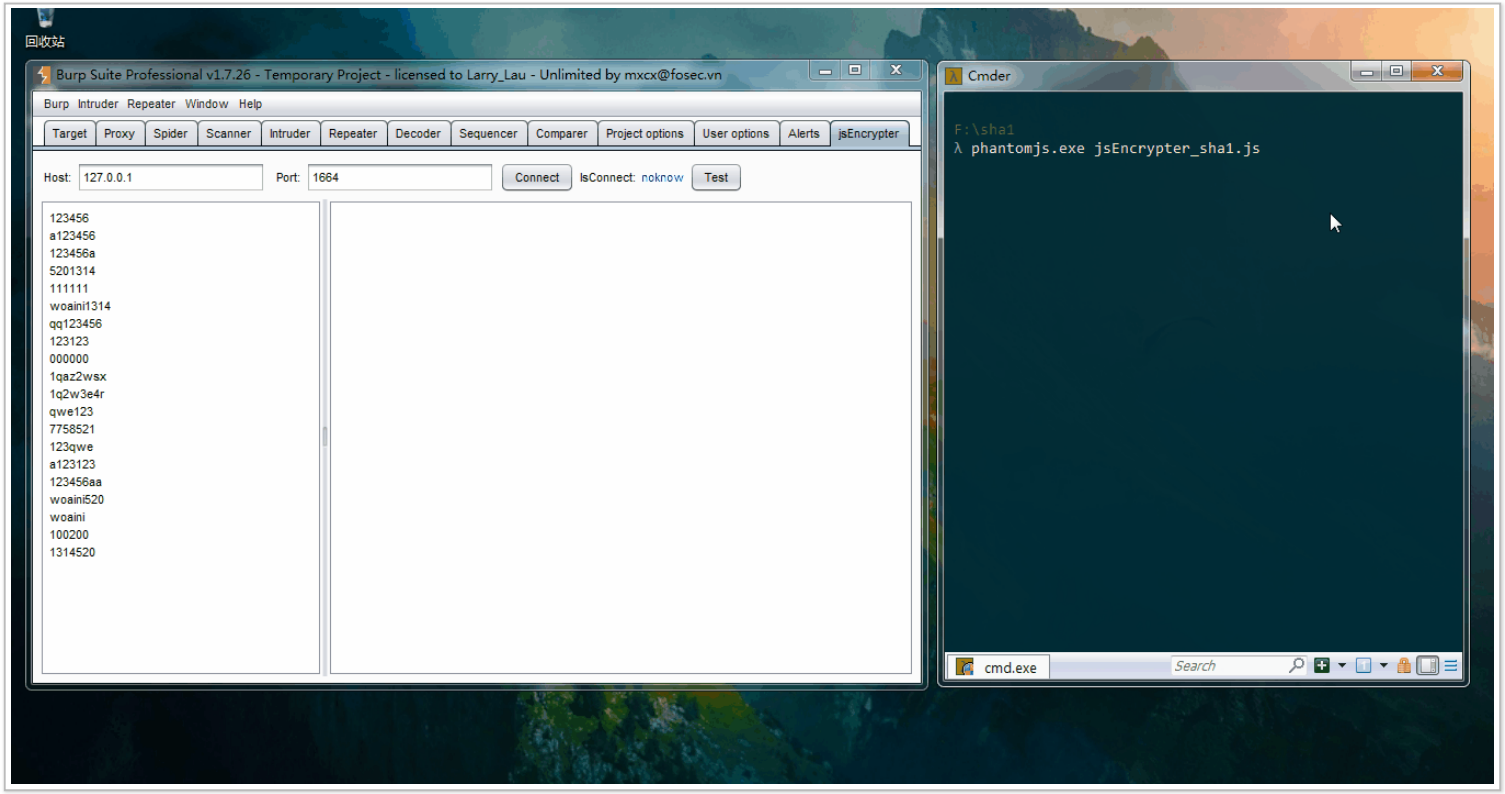
回收站
Burp Suite Professional v1.7.26 - Temporary Project - licensed to Larry_Lau - Unlimited by mxcx@fosec.vn
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Decoder Sequencer Comparer Project options User options Alerts jsEncrypter
Host: 127.0.0.1 Port: 1664 Connect IsConnect: noknow Test
123456
a123456
123456a
5201314
111111
woaini1314
qq123456
123123
000000
1qaz2wsx
1q2w3e4r
qwe123
7758521
123qwe
a123123
123456aa
woaini520
woaini
100200
1314520
Cmder
F:\sha1
λ phantomjs.exe jsEncrypter_sha1.js
cmd.exe
Search

图 4 - 测试

(5) 抓包爆破

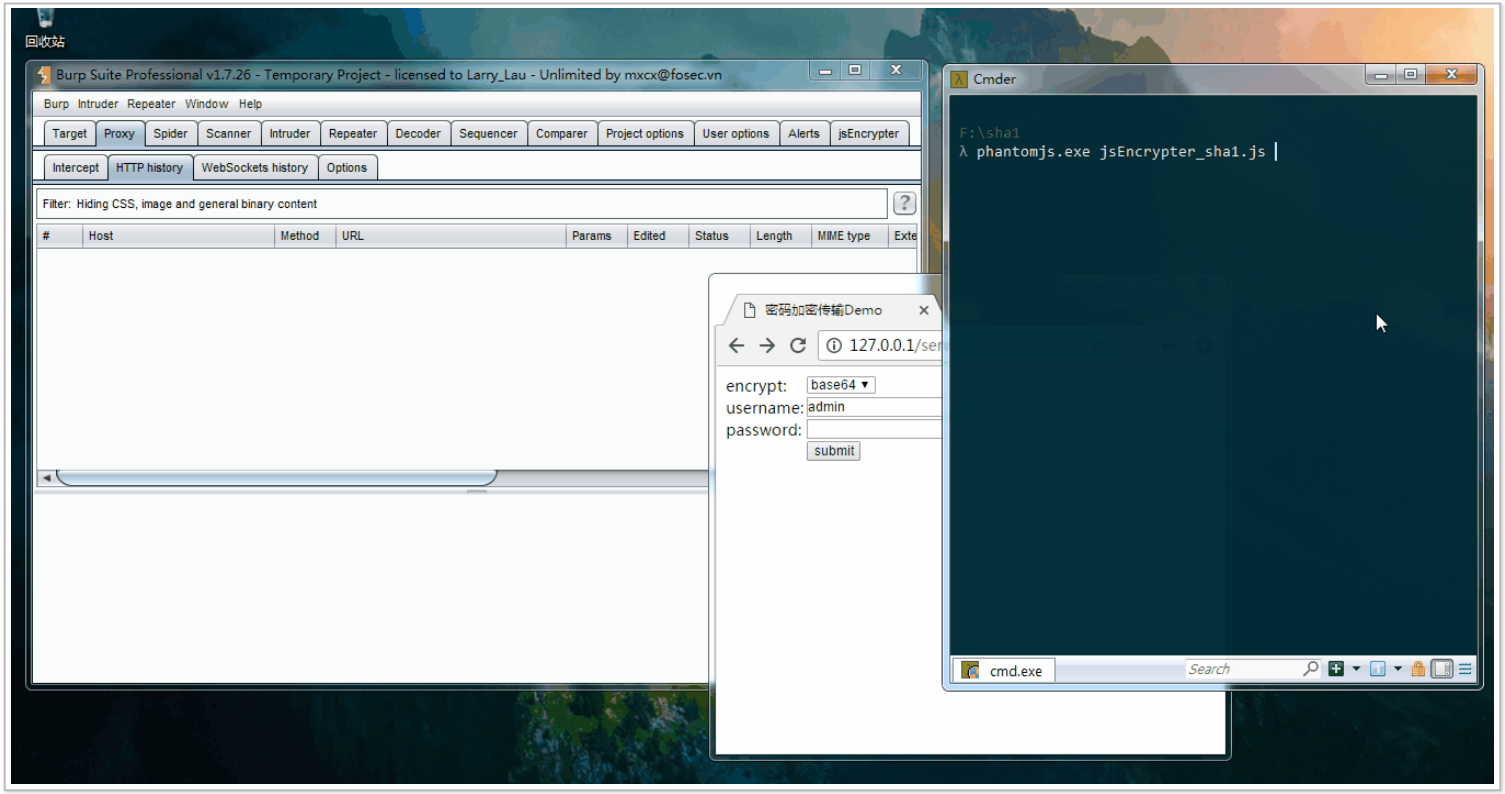

图 5 - 抓包爆破

0x04 最后的话

各位如果有更好的解决方案，请留言互相交流。发现项目有 bug 或者有更好的修改建议，欢迎在 github 提交 issuse，期待我们一起进步！

项目地址： https://github.com/c0ny1/jsEncrypter

全文完

本文由 简悦 SimpRead 优化，用以提升阅读体验。