Darkside RaaS in Linux version

cybersecurity.att.com/blogs/labs-research/darkside-raas-in-linux-version

1. AT&T Cybersecurity
2. Blog

June 22, 2021 | Ofer Caspi

Executive summary

AT&T Alien Labs recently analyzed the Linux version of the Darkside ransomware, one of the most active ransomware in the last quarter. Shortly after hitting Colonial Pipeline, Darkside developers announced they would be closing operations.

Key Points:

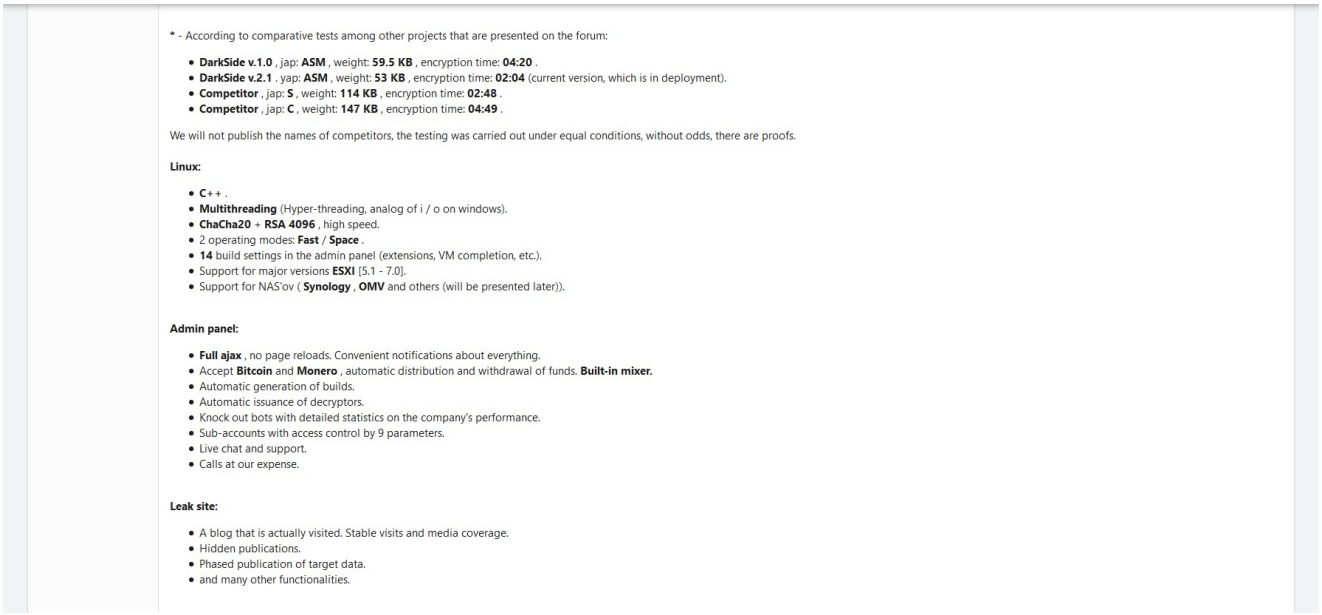
> Unlike common Linux ransomwares which mostly zip files with a password, Darkside encrypts files using crypto libraries. This likely makes recovery impossible without the encryption key, if properly implemented.

Background

Linux and UNIX servers have always been a preferred option for servers and data centers, likely due to the small attack surface of the servers, tight configurations, and lack of user interaction. However, they are often set up and then forgotten, left without detection or

protection mechanisms. This makes them very attractive to attackers. By infecting unprotected virtualization servers, attackers can perform devastating attacks on companies, taking down all the services of a company with a single infection.

First discovered in August 2020, Darkside is a group that operates ransomware as a service (RaaS), and attacks by their network of affiliates have infected many companies worldwide. The most high profile infection happened when a Darkside affiliate hit the network of one of the major oil pipeline companies in the US, Colonial Pipeline, causing the company to shut down operations for days. After this attack — likely due to the fact that it drew unwanted levels of attention to the group globally — the malware authors announced publicly the closure of their service. Nevertheless, there is evidence that the group has completed a Linux version of its malware that is targeting ESXi, servers hosting VMware virtual machines. To this point, the authors announced the Darkside 2.0 version with Linux capabilities on March 9, 2021 in the XSS Forum:



* - According to comparative tests among other projects that are presented on the forum:

- **DarkSide v.1.0** , jap: **ASM** , weight: **59.5 KB** , encryption time: **04:20** .
- **DarkSide v.2.1** , yap: **ASM** , weight: **53 KB** , encryption time: **02:04** (current version, which is in deployment).
- **Competitor** , jap: **S** , weight: **114 KB** , encryption time: **02:48** .
- **Competitor** , jap: **C** , weight: **147 KB** , encryption time: **04:49** .

We will not publish the names of competitors, the testing was carried out under equal conditions, without odds, there are proofs.

Linux:

- **C++** .
- **Multithreading** (Hyper-threading, analog of i / o on windows).
- **ChaCha20** + **RSA 4096** , high speed.
- 2 operating modes: **Fast** / **Space** .
- **14** build settings in the admin panel (extensions, VM completion, etc.).
- Support for major versions **ESXI** [5.1 - 7.0].
- Support for NAS'ov (**Synology** , **OMV** and others (will be presented later)).

Admin panel:

- **Full ajax** , no page reloads. Convenient notifications about everything.
- Accept **Bitcoin** and **Monero** , automatic distribution and withdrawal of funds. **Built-in mixer.**
- Automatic generation of builds.
- Automatic issuance of decryptors.
- Knock out bots with detailed statistics on the company's performance.
- Sub-accounts with access control by 9 parameters.
- Live chat and support.
- Calls at our expense.

Leak site:

- A blog that is actually visited. Stable visits and media coverage.
- Hidden publications.
- Phased publication of target data.
- and many other functionalities.

Figure 1. Example of Linux version features and information announced by malware authors on XSS Forum.

Analysis

Unlike the Windows version of the malware that targets any Windows endpoint, Darkside Linux version is mostly targeting ESXi servers. Its default configuration includes the root path of ESX server machines. Targeted extensions are 'vmdk', 'log', 'vmem', 'vmsn' that are used in ESX servers for saving virtual machines information, data, and logs.

The malware is quite informative and prints to the screen most of the actions it performs, which is an uncommon behavior for malware. This could imply that the malware is being deployed manually. The malware is written in C++ and it's also using several open-source libraries that were imported and compiled with the malware code into one binary. Some of

these libraries are: crypto++ used for encryption, boost used for logging purposes, and curl used for HTTP requests. The use of these libraries enables the final binary to be a 2.7MB file size. For example, to communicate with an infected machine with its Command and Control (C&C), the malware uses libcurl functions that were compiled with the rest of the code.

In addition, the malware supports command line parameters during execution to replace almost, if not all, of the default configuration variables.

```
v32 = sub_456CD0(v173, "help,h", "Help Screen");
v33 = sub_4571D0(v32, "size,s", v140, "Part Size to Process");
v34 = sub_4571D0(v33, "space,S", v139, &unk_5BC761);
v35 = sub_4571D0(v34, "dir,d", v21, "Root Directory Path to Process");
v36 = sub_4571D0(v35, "ext,x", v138, "Extension To Apply For Renaming");
v37 = sub_4571D0(v36, "new,n", v137, "Extension To Apply For Encrypted files");
v38 = sub_4571D0(v37, "log,l", v135, "Log File Path");
v39 = sub_4571D0(v38, "thread,t", v132, "Worker Threads Count, 0 - dynamic");
v40 = sub_4571D0(v39, "key,k", v143, "RSA Public Key File Paths");
v41 = sub_4571D0(v40, "rc2,e", v142, "RC2 Key as HEX string");
v42 = sub_4571D0(v41, "content,c", v141, "ReadMe File Path");
sub_4571D0(v42, "readme,r", v8, "ReadMe File name");
```

Figure 2. Parameters supported by the malware.

The malware also supports shutting down virtual machines by executing the esxcli commands, a special console on ESX servers that allows it to interact with virtual machines from the command line.

```
byte_8A3538 = 1;
_mm_mfence();
std::string::string(v1, "/sbin/esxcli", v2);
sub_46BA00(v2, v1, 0LL);
byte_8A3530 = v2[0] == 2;
```

```
std::string::string(v14, "--world-id=", &v25);
std::string::append((std::string *)v14, a2);
std::string::string(&v25, "vm", &v13);
std::string::string(&v26, "process", v15);
std::string::string(&v27, "kill", v16);
std::string::string(&v28, "--type=force", v20);
```

Figure 3. Abusing esxcli command.

When executed, the malware prints its configuration to the terminal. This includes the root path to encrypt, RSA key information, targeted file extensions to encrypt, C2 addresses, and more, as seen in figure 4.

```
[CFG] Root Path................/vmfs/volumes/
[CFG] Key Size.................548 Bytes
[CFG] Public Key..............VALID
[CFG] Part Size................500mb
[CFG] Space Size...............0mb
[CFG] Min Size.................1mb
[CFG] Search Extension.........vmdk,vmem,vswp,log,vmsn
[CFG] New Extension............darkside
[CFG] Thread Count.............1
[CFG] ReadMe File..............darkside_readme.txt
[CFG] ReadMe Size..............1969 Bytes
[CFG] Landing URL#[01].........http://catsdegree.com/bdbddbbeaed
[CFG] Landing URL#[02].........http://temisleyes.com/abbacbcd
[CFG] User ID..................46017379a796803
[CFG] RC2 Key..................OK

[INF] Scanning: /vmfs/volumes/
```

Figure 4. Darkside configuration.

The C&C addresses are encrypted using a rotated XOR key, which will be decrypted when the malware is executed. The decrypted addresses can be found in Appendix A:

```
90          do
91          {
92            v24 = *v5;
93            if ( *v5 )
94            {
95              key_byte = xor_key[key_index];
96              if ( v24 != key_byte )
97              {
98                ++key_index;
99                *v5 = key_byte ^ v24;
100               if ( key_index == 32 )
101                 key_index = 0LL;
102             }
103           }
104           ++v5;
105         }
```

Figure 5. Decryption loop to extract C&C.

```
xor_key         db 41h, 15h, 49h, 16h, 20h, 78h, 2Ch, 28h, 1Fh, 59h, 42h
                                        ; DATA XREF: oc_check_vmfs_path+14↑o
                                        ; oc_get_landing_url+25F↑r
                db 71h, 42h, 18h, 65h, 67h, 65h, 15h, 43h, 33h, 70h, 24h
                db 78h, 30h, 3Ch, 3Dh, 18h, 78h, 6Fh, 22h, 29h, 48h
```

Figure 6. Malware XOR key.

The malware will then count the files to be encrypted, and it collects information from the infected machine, sending it to the C&C server after encryption. The exfiltrated information includes: user name, OS version, hostname, build, and more:

```
oc_str_init((std::string *)host_name_s, (__int64)"hostname");
sub_438890((unsigned int)v117, (unsigned int)host_name_s, (unsigned int)&machine_host_name, v9, v10, v11, v88);
std::string::_Rep::_M_dispose(host_name_s[0] - 24, &name);
std::string::_Rep::_M_dispose(machine_host_name - 24, &name);
oc_str_init((std::string *)domain_s, (__int64)"domain");
std::locale::locale((std::locale *)v116);
std::locale::locale((std::locale *)&name, (const std::locale *)v116);
sub_446CF0(v117, domain_s, 6170175LL, &name);
std::locale::~locale(&name);
std::locale::~locale(v116);
std::string::_Rep::_M_dispose(domain_s[0] - 24, &name);
oc_str_init((std::string *)s_version, (__int64)"version");
sub_438890((unsigned int)v117, (unsigned int)s_version, (unsigned int)&ma_version, v12, v13, v14, v89);
std::string::_Rep::_M_dispose(s_version[0] - 24, &name);
uid = (pthread_rwlock_t *)byte_8A2478;
v160 = byte_8A2478;

oc_get_uid(a2, &uid, &v160);

oc_str_init((std::string *)s_username, (__int64)"username");
sub_438890((unsigned int)v117, (unsigned int)s_username, (unsigned int)&uid, v15, v16, v17, v90);
std::string::_Rep::_M_dispose(s_username[0] - 24, &name);
oc_str_init((std::string *)group, (__int64)"group");
```

Figure 7. Collecting system information.

Figure 8 shows an example of the information collected from an infected machine.

```
.................r.......r.......r.......r..............E.............................wncuejeh=.1.eric................1.......
.............................d":·"cbba22b2baaaba12cac2",·····"uid":·"46017379a796803",·····"hostname":·"ubuntu",·····"domain"
:·"-",·····"version":·"1.0",·····"username":·"ofer",·····"group":·"1000",·····"os_type":·"Linux",·····"os_version":·"Linux·#46-U
buntu·SMP·Fri·Jul·10·00:24:02·UTC·2020",·····"os_build":·"5.4.0-42-generic",·····"os_arch":·"x86_64",·····"disks":·[·········{··
···········"MountPoint":·"\/",·············"Type":·"ext4",·············"Device":·"\/dev\/sda5",·············"Size":·"19523",·····
·········"Available":·"2668",·············"Free":·"3682".·····},·········{·············"MountPoint":·"\/boot\/efi",·····
···"Type":·"vfat",·············"Device":·"\/dev\/sda1",·············"Size":·"510",·············"Available":·"510",·············"
Free":·"510".·····}.····].}.....................................................................................................
.....................................................................................................
.............................0.......4.............................{.····"id":·"cbba22b2baaaba12cac2",·····"uid":·"46017379
a796803",·····"hostname":·"ubuntu",·····"domain":·"-",·····"version":·"1.0",·····"username":·"ofer",·····"group":·"1000",·····"o
s_type":·"Linux",·····"os_version":·"Linux·#46-Ubuntu·SMP·Fri·Jul·10·00:24:02·UTC·2020",·····"os_build":·"5.4.0-42-generic",·····
·"os_arch":·"x86_64",·····"disks":·[·········{·············"MountPoint":·"\/",·············"Type":·"ext4",·············"Device":
·"\/dev\/sda5",·············"Size":·"19523",·············"Available":·"2668",·············"Free":·"3682".·····},·········{···
···········"MountPoint":·"\/boot\/efi",·············"Type":·"vfat",·············"Device":·"\/dev\/sda1",·············"Size":·"510
",···········"Available":·"510",·············"Free":·"510".·····}.····].}.....................................................
```

Figure 8. Example of collected system information.

The malware will loop through the files to be encrypted and then encrypt them using ChaCha20 with the RSA 4096 key taken from the configuration. After encryption, the malware will add a tail at the end of the encrypted file that includes a constant and the cipher, as seen in figure 9.

```
*(_QWORD *)constant = 0x912301DE00A122AALL;
v24 = v5;
std::ostream::write((std::ostream *)&file_des, constant, 12LL);
if ( v37 )
{
  v13 = (std::runtime_error *)__cxa_allocate_exception(0x20uLL);
  std::string::string(&v18, "Writing Header Failed", v22);
  sub_5B4100();
  __errno_location();
  sub_416B60(v13, v16, v17, v18, v19, v20, v21);
  std::string::_Rep::_M_dispose(v18 - 24, v30);
  __cxa_throw(v13, (struct type_info *)&`typeinfo for'std::system_error, sub_5B4050);
}
std::ostream::write((std::ostream *)&file_des, cipher, v29 - (_QWORD)cipher);
if ( v37 )
{
  v12 = (std::runtime_error *)__cxa_allocate_exception(0x20uLL);
  std::string::string(&v20, "Cipher Writing Failed", v22);     thrown_size: unsigned __int64
  sub_5B4100();
  __errno_location();
  sub_416B60(v12, v16, v17, v18, v19, v20, v21);
  std::string::_Rep::_M_dispose(v20 - 24, v30);
  __cxa_throw(v12, (struct type_info *)&`typeinfo for'std::system_error, sub_5B4050);
}
v6 = 0LL;
```

Figure 9. Adding header after successful encryption

After encryption, the malware creates a ransom note in each folder where files were encrypted.

```
1  ----------- [ Welcome to DarkSide ] ------------->
2
3  What happend?
4  ---------------------------------------------
5  Your computers and servers are encrypted, backups are deleted. We use strong encryption algorithms, so you
   cannot decrypt your data.
6  But you can restore everything by purchasing a special program from us - universal decryptor. This program will
   restore all your network.
7  Follow our instructions below and you will recover all your data.
8
9  What guarantees?
10 ---------------------------------------------
11 We value our reputation. If we do not do our work and liabilities, nobody will pay us. This is not in our
   interests.
12 All our decryption software is perfectly tested and will decrypt your data. We will also provide support in case
   of problems.
13 We guarantee to decrypt one file for free. Go to the site and contact us.
14
15 How to get access on website?
16 ---------------------------------------------
17 Using a TOR browser:
18 1) Download and install TOR browser from this site: https://torproject.org/
19 2) Open our website: http://darksidfqzcuhtk2.onion/-
   O7ADVDV5LR24AMM1KAYU5IJO4MMIBYA22RXI2K2EPNIAKZGNHUZNT933I6WBWPTN
20
21 When you open our website, put the following data in the input form:
22 Key:
23
24
   Nh2OzweFV9PHTGOmwd2gLkqLRZkYJ8WiGYK7NUw7pCuMEhiL3sdP6ctBxYHxfHuIMmFOscLNNe8N2w7lKKTsw2Safmkz95mEbPXVIFRdpCibu0CcE3
25
26 !!! DANGER !!!
27 DO NOT MODIFY or try to RECOVER any files yourself. We WILL NOT be able to RESTORE them.
28 !!! DANGER !!!
```

Figure 10. Ransom note

Recommended actions

1. Keep software up to date with security updates.
2. Carefully monitor and manage suspicious emails.
3. Use a backup system to backup server files.
4. Install Antivirus and/or endpoint detection and response (EDR) in all endpoints.
5. Make sure two-factor authentication is enabled in all services.

Conclusion

Ransomwares remains one of the biggest threats to companies globally, especially when it comes to virtual machine servers that may contain multiple machines that are primary targets for Darkside malware.

Darkside will search files on the main folder of the infected server and encrypt any file that matches its configuration limits. It will also collect information from the system and send it to its C&C.

Detection methods

The following associated detection methods are in use by AT&T Alien Labs. They can be used by readers to tune or deploy detections in their own environments or for aiding additional research.

YARA RULES

```
rule Darkside_linux

{

  meta:

        description = "darkside ransomware linux version"

        author = "Alienvault Labs"

        copyright = "Alienvault Inc. 2021"

  strings:

        $s1 = "[END] Remove Self"

        $s2 = "[CFG] Landing URL#["

        $s3 = "Welcome to DarkSide"

        $dec_loop = {0F B6 02 84 C0 74 1C 0F B6 B1 DF A7 89 00 40 38 F0 74 10 48 83
C1 01 31 F0 48 83 F9 20 88 02 49 0F 44 C8}


   condition:

        uint32(0) == 0x464C457F and all of them

}
```

Associated Indicators of Compromise (IOCs)

The following technical indicators are associated with the reported intelligence. A list of indicators is also available in the OTX Pulse. Please note, the pulse may include other activities related but out of the scope of the report.

TYPE	INDICATOR
DOMAIN	http://catsdegree[.]com/
DOMAIN	http://temisleyes[.]com/
DOMAIN	http://securebestapp20[.]com
SHA256	9cc3c217e3790f3247a0c0d3d18d6917701571a8526159e942d0fffb848acffb

SHA256	984ce69083f2865ce90b48569291982e786980aeef83345953276adfcbbeece8

SHA256	c93e6237abf041bc2530ccb510dd016ef1cc6847d43bf023351dce2a96fdc33b

Mapped to MITRE ATT&CK

The findings of this report are mapped to the following MITRE ATT&CK Matrix techniques:

- TA0043: Reconnaissance
  - TA1592: Gather Victim Host Information
- TA0042: Resource Development
  - T1583: Acquire Infrastructure
  - T1587: Develop Capabilities
- TA0040: Impact
  - T1486: Data Encrypted for Impact
- TA0005: Defense Evasion
  - T1027: Obfuscated Files or Information
- TA0007: Discovery
  - T1083: File and Directory Discovery
- TA0009: Collection
  - T1005: Data from Local System
- TA0011: Command and Control
  - T1001: Data Obfuscation
  - T1041: Exfiltration Over C2 Channel

Share this with others

Tags: malware, alien labs, otx, ransomware, labs, linux, threats, analysis, darkside raas