Ransomware

DarkSide on Linux: Virtual Machines Targeted

We focus on the behavior of the DarkSide variant that targets Linux. We discuss how it targets virtual machine-related files on VMware ESXI servers, parses its embedded configuration, kills virtual machines (VMs), encrypts files on the infected machine, collects system information, and sends it to the remote server.

By: Mina Naiim
May 28, 2021
Read time: 5 min (1371 words)

Subscribe

Updated June 1, 2021, 12:02 am ET: This article has been updated to remove the Command-and-Control (C&C) URI String field in Table 1. Further study showed that it does not apply consistently to a number of samples.

As we discussed in our previous blog, the DarkSide ransomware is targeting organizations in manufacturing, finance, and critical infrastructures in regions such as the United States, France, Belgium, and Canada. The DarkSide ransomware targets

In this blog, we focus on the behavior of the variant that targets Linux. This entry also discusses how this variant targets virtual machine-related files on VMware ESXI servers, parses its embedded configuration, kills virtual machines (VMs), encrypts files on the infected machine, collects system information, and sends it to the remote server.

This table summarizes some of the differences between the behavior of the DarkSide ransomware on Windows and on Linux:

Table 1. Comparison of DarkSide variants on Windows and Linux

	Windows Variant	Linux Variant
Encryption Mechanism	Salsa20 with RSA-1024	ChaCha20 with RSA-4096
Cipher Blocks	Salsa20 matrix is custom and randomly generated using **"RtlRandomExW"**	ChaCha20 initial block is standard, built using **"expand 32-byte k"** as a constant string
Configuration	Encrypted	Not encrypted
Terminates VMs?	No	Yes
Target Files	All files on the system except the files, folders, and file extensions mentioned in the configuration	VM-related files on VMware ESXI servers, with specific file extensions mentioned in the configuration
New Extension	Generated by applying CRC32 several times on the HWID of the victim machine as **".4731c768"**	Hard-coded in the embedded configuration as **".darkside"** or passed by execution parameters
Ransom Note File Name	Consists of hard-coded part in the configuration as **"README."** and the generated ID mentioned previously; for example, **"README. 4731c768.TXT"**	Hard-coded in the embedded configuration as **"darkside_readme.txt"** or passed by execution parameters

As we noted earlier, DarkSide also has a Linux variant to infect more machines and cause more damage in the victim network. However, this variant is quite specific, as its main configuration targets VM-related files on VMware ESXI servers as seen in the following figure:

```
00000000004231F1 mov     esi, offset aVmdkVmemVswpLo ; "vmdk,vmem,vswp,log,vmsn"
00000000004231F6 mov     rbx, rdi
00000000004231F9 sub     rsp, 10h
00000000004231FD lea     rdx, [rsp+18h+var_9]
0000000000423202 call    __ZNSsC1EPKcRKSaIcE ; std::string::string(char const*,std::allocator<char> const&)
0000000000423207 add     rsp, 10h
000000000042320B mov     rax, rbx
```

Figure 1. Target file extensions

Configuration

Unlike the Windows variant, the Linux variant's strings and configuration are not obfuscated. The configuration of the Linux variant specifies features of the sample, such as the extension for encrypted files, C&C URL, number of threads, and a constraint on a minimum size of the target files to be encrypted.

Note that the root path — the starting point for encryption — in the following figure is "/vmfs/volumes/", which is the default location for the VM files on ESXI hosts.

```
[CFG] Part Size................500mb
[CFG] Space Size...............0mb
[CFG] Min Size.................1mb
[CFG] Search Extension.........vmdk,vmem,vswp,log
[CFG] New Extension............darkside
[CFG] Thread Count.............1
[CFG] ReadMe File..............darkside_readme.txt
[CFG] ReadMe Size..............1969 Bytes
[CFG] Landing URL#[01].........http://securebestapp20.com/daecdcbebcac
[CFG] User ID..................75fb1970b674cc4
[CFG] RC2 Key..................OK
```

Figure 2. Configuration of the Linux variant

In addition to the hard-coded configuration, the ransomware executable can accept parameters to infect more files and change its default settings. Figure 3 shows where the malware parses execution parameters.

```
v27 = sub_4309E0(0LL);
v28 = sub_42F370(v27, &v156);
v153 = sub_455A20(&v158);
v29 = sub_456CD0(&v153, "help,h", "Help Screen");
v30 = sub_4571D0(v29, "size,s", v28, "Part Size to Process");
v31 = sub_4571D0(v30, "space,S", v26, &unk_5BC761);
v32 = sub_4571D0(v31, "dir,d", v24, "Root Directory Path to Process");
v33 = sub_4571D0(v32, "ext,x", v22, "Extension To Apply For Renaming");
v34 = sub_4571D0(v33, "new,n", v20, "Extension To Apply For Encrypted files");
v35 = sub_4571D0(v34, "log,l", v18, "Log File Path");
v36 = sub_4571D0(v35, "thread,t", v16, "Worker Threads Count, 0 - dynamic");
v37 = sub_4571D0(v36, "key,k", v14, "RSA Public Key File Paths");
v38 = sub_4571D0(v37, "rc2,e", v12, "RC2 Key as HEX string");
v39 = sub_4571D0(v38, "content,c", v10, "ReadMe File Path");
sub_4571D0(v39, "readme,r", v8, "ReadMe File name");
```

Figure 3. Linux variant parameter parsing

ESXCLI Commands

DarkSide runs several ESXCLI commands (such as the command-line interface framework in vSphere) in order to collect information about the infected ESXI host,

Table 2 shows a list of ESXCLI commands run by DarkSide on the victim machine.

Table 2. ESXCLI Commands

Commands	Desription
esxcli --formatter=csv --format-param=fields=="Device,DevfsPath" storage core device list	List the Devfs Path of the devices currently registered with the storage
esxcli --formatter=csv storage filesystem list	List the logical sections of storage currently connected to the ESXI host
esxcli --format-param=fields=="WorldID,DisplayName" vm process list	List the running VMs on the ESXI host
esxcli vsan debug vmdk list	List the status of VMDKs in vSAN
esxcli --format-param=fields=="Type,ObjectUUID,Configuration" vsan debug object list	List the UUID of the vSAN objects

Figure 4 shows how the DarkSide ransomware lists the running virtual machines on the ESXI.

```
0000000000427DA5    mov     esi, offset aVm ; "vm"
0000000000427DAA    call    __ZNSsC1EPKcRKSaIcE ; std::string::string(char const*,std::allocator<char> const&)
0000000000427DAF    lea     rdi, [rbx+18h]
0000000000427DB3    lea     rdx, [rsp+98h+var_7C]
0000000000427DB8    mov     esi, offset aProcess ; "process"
0000000000427DBD    call    __ZNSsC1EPKcRKSaIcE ; std::string::string(char const*,std::allocator<char> const&)
0000000000427DC2    lea     rdi, [rbx+20h]
0000000000427DC6    lea     rdx, [rsp+98h+var_7B]
0000000000427DCB    mov     esi, offset aList ; "list"
0000000000427DD0    call    __ZNSsC1EPKcRKSaIcE ; std::string::string(char const*,std::allocator<char> const&)
0000000000427DD5    lea     rcx, [rsp+98h+var_7A]
```

Figure 4. Listing running VMs

Killing Virtual Machines

Before encryption, the Linux variant of the DarkSide ransomware can power off running VMs on the ESXI server using the following ESXI command:

```
“esxcli vm process kill --type= force --world-id= <WorldNumber>”
```

```
0000000000426FA9    mov     esi, offset aWorldId ; "--world-id="
0000000000426FAE    push    r12
0000000000426FB0    push    rbp
0000000000426FB1    mov     rbp, rdi
0000000000426FB4    push    rbx
0000000000426FB5    sub     rsp, 0F8h
0000000000426FBC    lea     r15, [rsp+128h+var_58]
0000000000426FC4    lea     rdi, [rsp+128h+var_108]
0000000000426FC9    mov     rdx, r15
0000000000426FCC    call    __ZNSsC1EPKcRKSaIcE ; std::string::string(char const*,std::allocator<char> const&)
0000000000426FD1    lea     rax, [rsp+128h+var_E8]
0000000000426FD6    lea     rdi, [rsp+128h+var_108] ; this
0000000000426FDB    mov     rsi, r13 ; std::string *
0000000000426FDE    mov     [rsp+128h+var_120], rax
0000000000426FE3    call    __ZNSs6appendERKSs ; std::string::append(std::string const&)
0000000000426FE8    lea     rax, [rsp+128h+var_E8]
0000000000426FED    lea     rdx, [rsp+128h+var_109]
0000000000426FF2    mov     esi, offset aVm ; "vm"
0000000000426FF7    mov     rdi, r15
0000000000426FFA    mov     [rsp+128h+var_120], rax
0000000000426FFF    call    __ZNSsC1EPKcRKSaIcE ; std::string::string(char const*,std::allocator<char> const&)
0000000000427004    lea     rax, [rsp+128h+var_E8]
0000000000427009    lea     rdi, [r15+8]
000000000042700D    lea     rdx, [rsp+128h+var_F8]
0000000000427012    mov     esi, offset aProcess ; "process"
0000000000427017    mov     [rsp+128h+var_120], rax
000000000042701C    call    __ZNSsC1EPKcRKSaIcE ; std::string::string(char const*,std::allocator<char> const&)
0000000000427021    lea     rax, [rsp+128h+var_E8]
0000000000427026    lea     rdi, [r15+10h]
000000000042702A    mov     esi, offset aKill ; "kill"
000000000042702F    mov     rdx, rax
0000000000427032    mov     [rsp+128h+var_120], rax
```

Figure 5. Terminating running VMs

```
00000000004378CF                mov     eax, [rax+8]
00000000004378D2                lea     rdi, [rbx+8]
00000000004378D6                mov     esi, offset aEsxiKillVms___ ; "[ESXi] Kill VMs................"
00000000004378DB                mov     [rsp+0A8h+var_58], eax
00000000004378DF                call    sub_418930
00000000004378E4                lea     rdi, [rbx+70h]
00000000004378E8                mov     esi, r13d
00000000004378EB                call    __ZNSolsEi ; std::ostream::operator<<(int)
00000000004378F0                mov     rdi, rbp
```

Figure 6. Reporting on VM killing status

Encryption

The Linux variant of the DarkSide ransomware uses a ChaCha20 stream cipher with RSA-4096 to encrypt targeted files on the victim machine.

It loops across the files on the root path mentioned in the embedded configuration or in the given parameter, as shown in Figure 7.

```
  v4 = opendir(*a2);
  v5 = v4;
  if ( v4 )
  {
LABEL_2:
    while ( 1 )
    {
      v6 = readdir(v5);
      if ( !v6 )
        break;
      while ( 1 )
      {
        v7 = v6->d_name;
        if ( !memcmp(v6->d_name, ".", 2uLL) || !memcmp(v6->d_name, "..", 3uLL) )
          break;
        v8 = v6->d_type;
        if ( v8 == 4 )
        {
          v11 = byte_8A2478;
          std::string::assign((std::string *)&v11, (const std::string *)v3);
          std::string::append((std::string *)&v11, "/");
          std::string::append((std::string *)&v11, v7);
          sub_435B80(v2, (const char **)&v11);
          goto LABEL_2;
```


Figure 7. Linux variant looping across files/directories

parameters.

```
v33 = std::operator<<<std::char_traits<char>>(v2 + 296, "[INFO] ");
v34 = (std::ostream *)std::operator<<<std::char_traits<char>>(v33, "File Size................");
v35 = sub_5B0CB0(v34);
v36 = (std::ostream *)std::operator<<<std::char_traits<char>>(v35, "mb (");
v37 = sub_5B0CB0(v36);
v38 = std::operator<<<std::char_traits<char>>(v37, " Bytes)");
std::endl<char,std::char_traits<char>>(v38, " Bytes)", v39);
v40 = *(_QWORD *)(v2 + 120);
v41 = *(_QWORD *)(v2 + 128);
*(_QWORD *)(v2 + 152) = 0x100000LL;
v42 = v40 >> 20;
if ( v42 < v41 >> 20 )
{
  if ( v42 < *(_QWORD *)(v2 + 160) )
  {
    LODWORD(v52) = 2;
    v57 = &v52;
    v47 = (const char *)sub_418430(v38, " Bytes)");
    v48 = (__int64 **)&v53;
    sub_4178B0(&v53, v47, &v57);
    while ( v53 )
    {
      v49 = sub_418430(v48, v47);
      sub_418610(&v57, v49, &v53);
      v47 = "File Too Small, Ignored";
```

Figure 8. Linux variant performing a file size check

The malware then opens the target file, reads the content based on the part and space size given in the configuration or in the parameters, encrypts them, and writes to the file as shown in the following code:

```
      do
      {
        std::istream::read((std::istream *)&v107, v68, v11);// Read_file
        if ( v109 )
        {
          v54 = (std::runtime_error *)__cxa_allocate_exception(32LL);
          std::string::string(&v59, "File Reading Failed", &v84);
          sub_5B4100(&v59, "File Reading Failed");
          v55 = *__errno_location();
          sub_416B60(v54);
          std::string::_Rep::_M_dispose(v59 - 24, &v85);
          __cxa_throw(v54, &off_8991C0, sub_5B4050);
        }
        Encryption_routine_sub_510EE0(&v88, v71, v68, v69 - (_QWORD)v68);// Encryption_Routine
        v10 = v71;
```

Unlike the Windows variant that randomly generates its custom Salsa20 matrix by calling **"RtlRandomExW"** several times, the malware uses the standard constant **"expand 32-byte k**" in the Chacha20 cipher used to encrypt files on the victim machine, as shown in the next figure.

```
00007FF158DBA4A0  65 78 70 61 6E 64 20 33  32 2D 62 79 74 65 20 6B  expand·32-byte·k
00007FF158DBA4B0  4B 86 93 6C 50 E4 ED 94  67 03 86 CE B6 49 D0 E7  KåôlPSfög.å+¦I-t
00007FF158DBA4C0  2B 8D 00 AD 3A 8C 6B 33  FF A9 10 25 CB F6 4C DD  +...:îk3·¬.%-÷L¦
00007FF158DBA4D0  00 40 00 00 00 00 00 00  BA 3D F3 43 08 8D 6F 18  .@......¦==C..o.
00007FF158DBA4E0  00 01 00 00 00 00 00 00  FF FF FF FF FF FF FF 3F  ........·······?
00007FF158DBA4F0  10 00 00 00 00 00 00 00  A0 A4 DB 58 F1 7F 00 00  .........áñ¦X±...
00007FF158DBA500  14 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00007FF158DBA510  F8 2D 8A 00 00 00 00 00  70 2D 8A 00 00 00 00 00  °-è.....p-è.....
00007FF158DBA520  80 5E 00 4C F1 7F 00 00  80 5E 00 4C F1 7F 00 00  Ç^.L±...Ç^.L±...
00007FF158DBA530  80 5E 00 4C F1 7F 00 00  80 5E 00 4C F1 7F 00 00  Ç^.L±...Ç^.L±...
00007FF158DBA540  80 5E 00 4C F1 7F 00 00  7F 7E 00 4C F1 7F 00 00  Ç^.L±....~.L±...
00007FF158DBA550  A0 C6 2E 5A F1 7F 00 00  00 00 00 00 00 00 00 00  á¦.Z±...........
00007FF158DBA560  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00007FF158DBA570  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00007FF158DBA580  60 2D 00 4C F1 7F 00 00  01 00 00 00 00 00 00 00  `-.L±...........
00007FF158DBA590  1C 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00007FF158DBA5A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00007FF158DBA5B0  80 5E 00 4C F1 7F 00 00  00 20 00 00 00 00 00 00  Ç^.L±....·......
00007FF158DBA5C0  01 00 01 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

Figure 10. Using "expand 32-byte k" as a constant in the Chacha20 cipher

After encryption, the malware then adds a header and a cipher at the end of the encrypted files as shown in Figure 11.

```
std::ostream::write((std::ostream *)&v41, v27, 12LL);
if ( v42 )
{
  v17 = (std::runtime_error *)__cxa_allocate_exception(32LL);
  std::string::string(&v24, "Writing Header Failed", &v26);
  sub_5B4100(&v24, "Writing Header Failed");
  v18 = *__errno_location();
  sub_416B60(v17);
  std::string::_Rep::_M_dispose(v24 - 24, &v34);
  __cxa_throw(v17, &off_8991C0, sub_5B4050);
}
std::ostream::write((std::ostream *)&v41, v32, v33 - (_QWORD)v32);
if ( v42 )
{
  v15 = (std::runtime_error *)__cxa_allocate_exception(32LL);
  std::string::string(&v25, "Cipher Writing Failed", &v26);
  sub_5B4100(&v25, "Cipher Writing Failed");
```

Figure 11. Adding code to header

Figure 12. Hex view of the encrypted file

The ransomware output console shows the results of the encryption, the encrypted filenames, the discarded files after size check, the time of encryption, and more.

```
[START #01] File Path.........../vmfs/volumes//here.log
[INFO] File Size...............0mb (10492 Bytes)
[ERROR] File Too Small, Ignored


[START #01] File Path.........../vmfs/volumes//test4.vmsn
[INFO] File Size...............7mb (8082169 Bytes)
[STOP] Elapsed Time............. 5644.908650s wall, 0.010000s user + 0.060000s
system = 0.070000s CPU (0.0%)
```


Figure 13. Ransomware output console

The Linux variant drops a ransom note on the victim machine and adds a new file extension to the encrypted files.

Unlike the Windows variant, the ransom note file name and the new extension for encrypted files are hard-coded in the malware configuration file or given in a parameter, and the malware does not add any ID at the end of it.

For the analyzed samples, the new extension was **".darkside**" and the hard-coded ransom note file name was **"darkside_readme.txt"**.

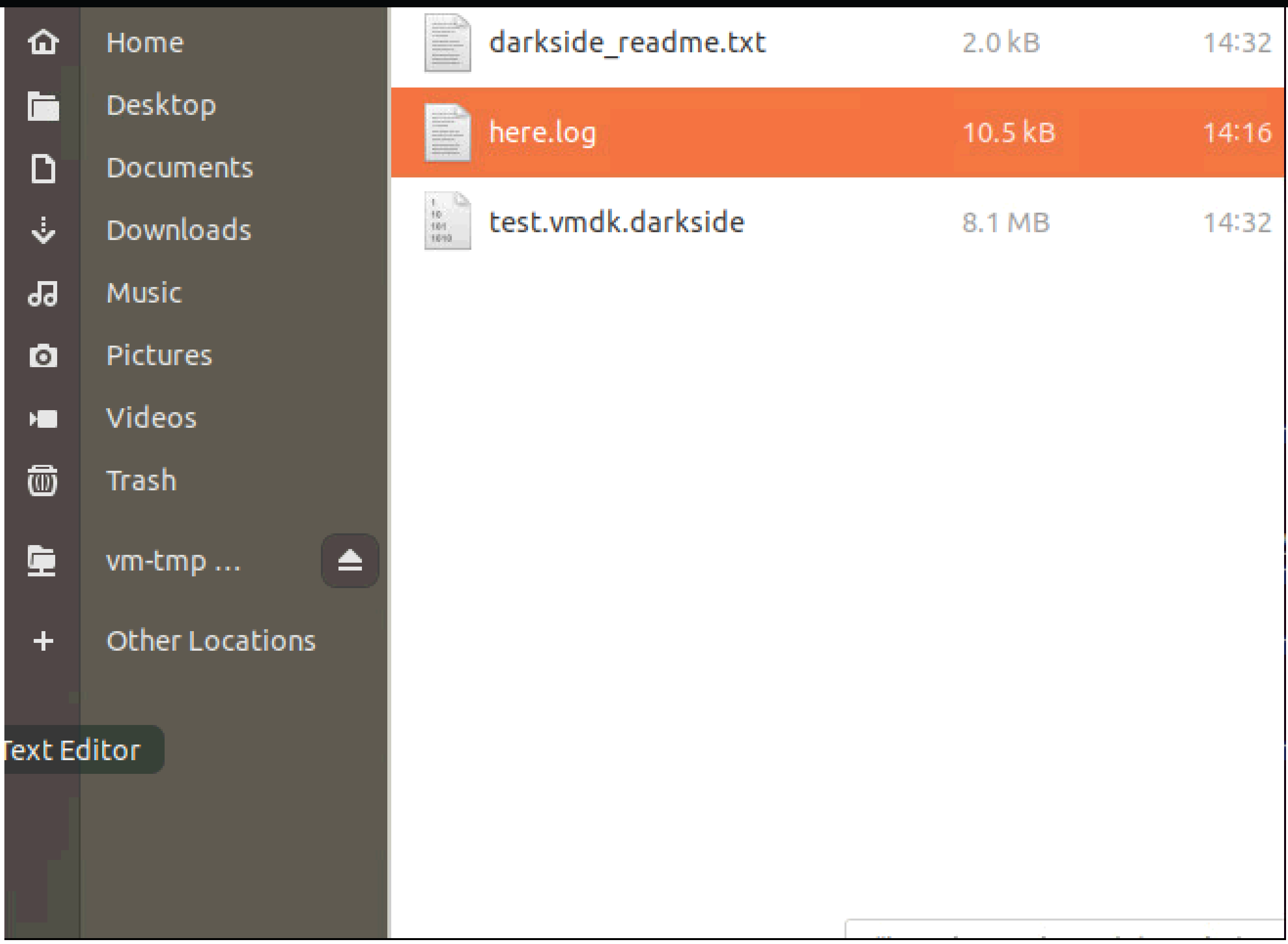

Figure 14. Encrypted folder with ransom note

C&C Beaconing

The DarkSide ransomware can send a C&C beaconing message with the collected system information to a remote server hardcoded in the configuration. It collects system information on the victim machine, such as host name, domain, and disk information, as evidenced in Figure 15.

```
  sub_417930(&v90);
}
LODWORD(v117) = 0;
*(_QWORD *)&name.sysname[0] = &v117;
v9 = sub_418430(v6, v7);
v10 = &v125;
v11 = (char *)v9;
sub_4178B0(&v125, v9, &name);
while ( v125 )
{
  v93 = sub_418430(v10, v11);
  v12 = sub_498940(&v125);
  v94 = v12;
  v12 += 8LL;
  v95 = *(_DWORD *)(__cxa_get_globals() + 8);
  sub_418930(v12, "group: ");
  v11 = v122;
  sub_418630(v12, v122, *((_QWORD *)v122 - 3));
  v10 = (__int64 **)&v93;
```

Figure 15. System information collection

The ransomware then puts the collected system information of the victim machine with a hard-coded UID value in the following format:

```
debug006:00007F5C200076C8 aIdAa21bbc2aa21abacab2cUid46017379a796803 db '{',0Ah
debug006:00007F5C200076C8 db '    "id": "aa21bbc2aa21abacab2c",',0Ah
debug006:00007F5C200076C8 db '    "uid": "46017379a796803",',0Ah
debug006:00007F5C200076C8 db '    "hostname": "ComputerNameUbuntu",',0Ah
debug006:00007F5C200076C8 db '    "domain": "-",',0Ah
debug006:00007F5C200076C8 db '    "version": "1.0",',0Ah
debug006:00007F5C200076C8 db '    "username": "username",',0Ah
debug006:00007F5C200076C8 db '    "group": "1000",',0Ah
debug006:00007F5C200076C8 db '    "os_type": "Linux",',0Ah
debug006:00007F5C200076C8 db '    "os_version": "Linux #42-Ubuntu SMP Tue Oct 23 15:48:01 UTC 2'
debug006:00007F5C200076C8 db '018",',0Ah
debug006:00007F5C200076C8 db '    "os_build": "4.15.0-39-generic",',0Ah
debug006:00007F5C200076C8 db '    "os_arch": "x86_64",',0Ah
debug006:00007F5C200076C8 db '    "disks": [',0Ah
debug006:00007F5C200076C8 db '        {',0Ah
debug006:00007F5C200076C8 db '            "MountPoint": "\/",',0Ah
debug006:00007F5C200076C8 db '            "Type": "ext4",',0Ah
debug006:00007F5C200076C8 db '            "Device": "\/dev\/sda1",',0Ah
debug006:00007F5C200076C8 db '            "Size": "50138",',0Ah
debug006:00007F5C200076C8 db '            "Available": "38679",',0Ah
debug006:00007F5C200076C8 db '            "Free": "41255"',0Ah
debug006:00007F5C200076C8 db '        }',0Ah
debug006:00007F5C200076C8 db '    ]',0Ah
debug006:00007F5C200076C8 db '}',0Ah,0
debug006:00007F5C2000790F db    0
```

Figure 16. System information format

eight characters in the request body to make its C&C traffic more difficult to detect by IPS/IDS devices on the victim network. The request body has the following format:

```
<Random 8-character variable> = <Encrypted collected information> &
<Random 8-character variable> = <hardcoded UID>
```

Figure 17 shows the HTTP POST request sent by the malware to the remote server with the collected information.

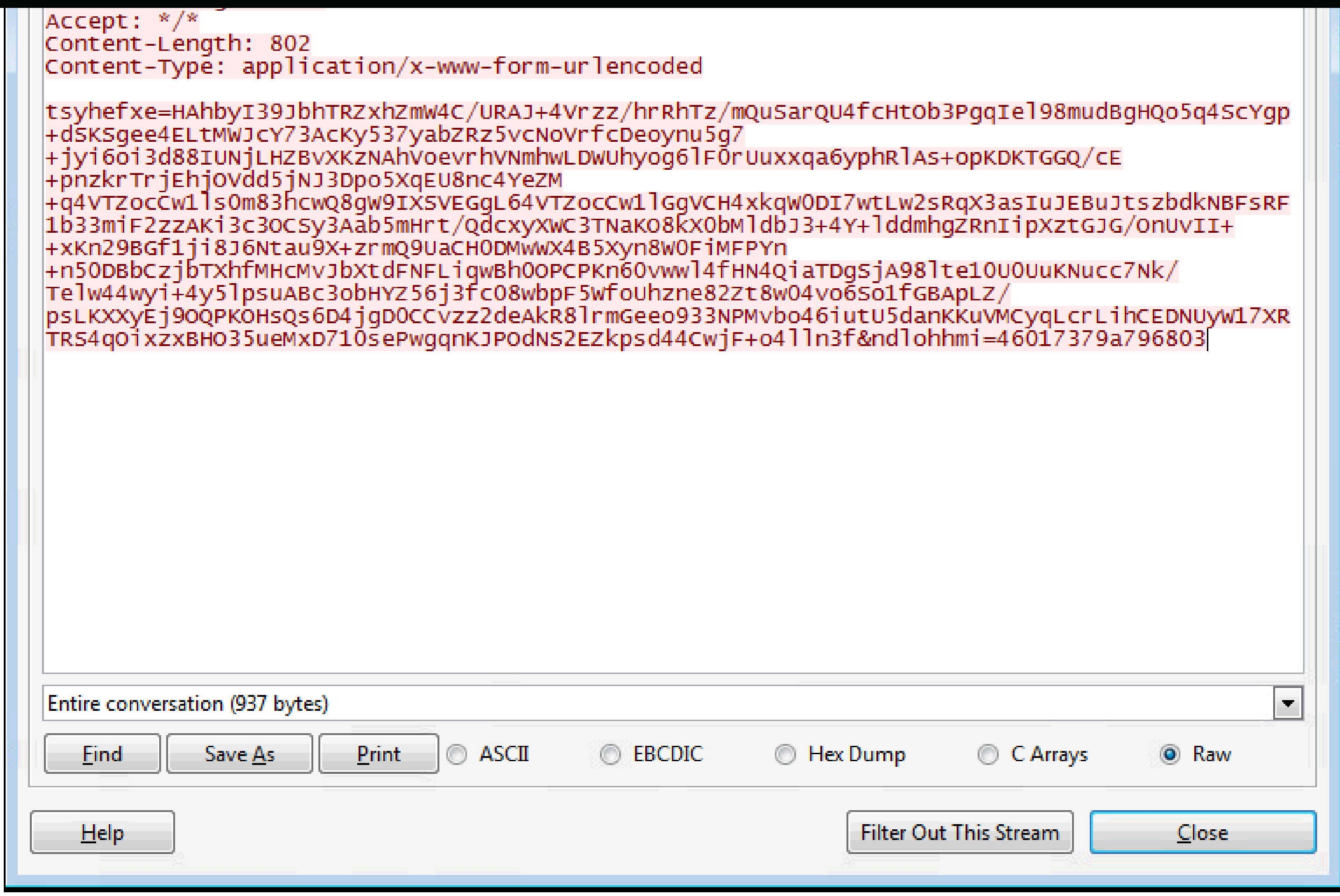

Figure 17. C2 beaconing HTTP traffic

Conclusion

The DarkSide ransomware family targets both Windows and Linux platforms. There are similarities between the Linux and Windows variants, but they are different with regard to some features, such as encryption mechanism, target files, ransom note name, extension, C&C URL, and more.

The Linux variant uses a ChaCha20 stream cipher with RSA-4096 in order to encrypt the

VMWare ESXI servers, such as VMDK files. It can also accept parameters to infect more files on the victim directories on the victim machine.

Indicators of Compromise

C&C servers:

- catsdegree[.]com
- securebestapp20[.]com
- temisleyes[.]com

SHA256	Trend Micro Detection Nam
984ce69083f2865ce90b48569291982e786980aeef83345953276adfcbbeece8	Ransom.Linux.DARKSIDE.THDE
9cc3c217e3790f3247a0c0d3d18d6917701571a8526159e942d0fffb848acffb	
c93e6237abf041bc2530ccb510dd016ef1cc6847d43bf023351dce2a96fdc33b	
da3bb9669fb983ad8d2ffc01aab9d56198bd9cedf2cc4387f19f4604a070a9b5	

Tags

Articles, News, Reports | Ransomware | Research

Mina Naiim
Threats Analyst

Related Articles

Understanding the Initial Stages of Web Shell and VPN Threats: An MXDR Analysis

Attacker Abuses Victim Resources to Reap Rewards from Titan Network

A Cybersecurity Risk Assessment Guide for Leaders

See all articles ›

TREND MICRO | Business

Resources

Support

About Trend

Country Headquarters

Trend Micro - United States (US)

225 East John Carpenter Freeway
Suite 1500
Irving, Texas 75062

Phone: +1 (817) 569-8900

Select a country / region

United States

Privacy | Legal | Accessibility | Site map