Malware Analysis Report: Locky Ransomware

- **Student Names:**
  - Karim Elamsry 221005999
  - Abdelrahaman 221006389
  - Haidy Ahmed 221006472

- **Date:** May 20, 2025
- **Course:** Malware Analysis and Reverse Engineering
- **Instructor:** Dr. Mohamed Elhamahmy
- **Sample Name:** SecuriteInfo.com.Trojan.Encoder.3976.32157.17259
- **Sample MD5:** c209817538e86f5ea49fa6bd180dbf01
- **Lab Environment:** FLARE VM (Windows 10, VMware Workstation, Host-Only Network), Any.run Sandbox

Table of Contents

Executive Summary

This report presents a comprehensive analysis of the Locky malware sample (SecuriteInfo.com.Trojan.Encoder.3976.32157.17259), a notorious ransomware. Key findings include:

- **Static Analysis:** The sample is packed by dynamic loading, uses obfuscated strings, and imports APIs for network communication and memory manipulation.
- **Reverse Engineering:** The code reveals heavy obfuscation, dynamic loading, API name hashing, and payload unpacking via RC4 decryption.
- **Dynamic Analysis:** Sandbox execution shows registry modifications, file enumeration, and communication with a C2 server.
- **Incident Response:** Recommended steps include isolating infected systems, blocking C2 IPs, and restoring from backups.

Analysis Methodology

The analysis was conducted in four phases, using tools pre-installed in FLARE VM and Any.Run environment:

Static Analysis

- **Tools:** PEiD, PeStudio,Detect it Easy ,FLOSS, Strings, Ghidra
- **Objective:** Examine the sample's structure, imports, strings, and code without execution.

Reverse Engineering

- **Tool:** Ghidra
- **Objective:** Disassemble and decompile the binary to understand its logic and obfuscation.

Dynamic Analysis

- **Tool:** Any.Run Sandbox, x32 debug, FlareVM, Wireshark, ProcMon, Fakenet
- **Objective:** Execute the sample in a controlled environment to observe its behavior.

Incident Response

- **Framework:** NIST SP 800-61
- **Objective:** Propose a response plan based on analysis findings.

Safety Measures

- Static analysis was performed in an isolated VirtualBox VM with no internet access.
- Dynamic analysis used Any.Run Sandbox and FlareVM in a controlled network.
- The sample was stored in a password-protected ZIP.

Static Analysis

PEiD

Tool: PEiD

Findings

- **Signature:** Microsoft Linker 14.0 | Microsoft Visual C++ 6.0 - 8.0 | Visual Studio 2013
- **Entropy:** 7.630 (high, suggesting obfuscation).
- **File Type:** executable, 32-bit, GUI (Windows PE32 executable).
- **Entry point:** 0x000098F0

Implications

Packing with unknown tools indicates Locky is trying to evade analysis and detection.

PeStudio & Detect it Easy Analysis

Tool: PeStudio
Sections: 5 sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5
.text	4096	77703	77824	6.86	c75380034a9494f7f51a8c05a5a5786
.rdata	81920	23694	24064	4.83	f88fda5b8b970deb86da6b315f159f43
.data	106496	5644	3072	3.06	ef4b746873c34e4b5e817f9f9c89ebd2
.rsrc	114688	952	1024	2.71	1667cd98cb72ec7d55e4f2c6f2e99f66
.reloc	118784	151840	152064	7.99	87069f395d685cbf701f5abde95ef63d

Packing

- Section 4 (.reloc) has high entropy and is unusually large, but no known packer is used. Unpacking must be done by debugging.
- Generic (packed and hide the most imports to do not detect that is malware) [No extension import]

Imports

- **GetEnvironmentStringsW, GlobalMemoryStatusEx, GetNumberOfConsoleInputEvents**: Reconnaissance; Sandbox/VM detection.
- **GetCurrentProcessId, GetCurrentThreadId, GetCurrentProcess, GetModuleHandleExW**: Execution setup; Context awareness
- **SystemFunction036, WriteFile**: Key generation and file encryption
- **RaiseException, IsDebuggerPresent**: Evasion; Debugger crash/control.
- **FindFirstFileExA, FindNextFileA**: Targeting; scanning for victim files

Libraries

- **ADVAPI32.dll** (Advanced Windows 32 Based API Called)
- **KERNEL32.dll** (Windows NT Base API Client)
- **Z** (Encrypted library) with 0 imports (that is new that the library is found without any imports)
- **dGetStdHandle** (Encrypted library)
- **§QueryPerformanceCounter** (Encrypted library) (highest imports with 321 imports)

Strings

API Set Name	Description	Likely Purpose in Malware
api-ms-win-security-systemfunctions-l1-1-0	Security-related functions, e.g., SystemFunction036 (RtlGenRandom) for cryptographic operations	Likely used to generate encryption keys or random data for obfuscation or ransomware encryption
api-ms-win-rtcore-ntuser-window-l1-1-0	Low-level user interface windowing operations (modern UWP-related)	Possibly used to manipulate or interact with UI components, or spoof user input
api-ms-win-core-xstate-l2-1-0	Extended processor state management, used in context switching	May be used to manage thread or CPU state — sometimes used in advanced shellcode
api-ms-win-core-winrt-l1-1-0	Windows Runtime (WinRT) support functions	Used for interacting with WinRT objects, possibly in modern Windows apps or obfuscated payloads
api-ms-win-core-sysinfo-l1-2-1	System information functions like OS version, memory, CPU count	May be used to fingerprint the environment or check for sandboxes/VMs
api-ms-win-core-synch-l1-2-0	Thread and process synchronization primitives (mutexes, semaphores)	Used to control concurrency or check for existing running instances of malware
api-ms-win-core-string-l1-1-0	String manipulation and safe string handling functions	May help decode, construct, or obfuscate command and control strings or paths

API Set Name	Description	Likely Purpose in Malware
api-ms-win-core-processthreads-l1-1-2	Process and thread management (e.g., CreateProcess, CreateThread)	Likely used for creating or injecting into processes (e.g., malware injection techniques)
api-ms-win-core-localization-obsolete-l1-2-0	Legacy localization and language support	Not directly malicious, but may be present from default system builds
api-ms-win-core-localization-l1-2-1	Localization and culture-specific data support	Could be used to tailor behavior to user's language or region
api-ms-win-core-file-l2-1-1	File system access and manipulation (open, read, write, delete)	Likely used to access or encrypt victim files (common in ransomware)
api-ms-win-core-fibers-l1-1-1	Fiber management for user-mode thread scheduling	Sometimes used for lightweight multithreading in malware payloads
api-ms-win-core-datetime-l1-1-1	Date and time handling functions	Used for time-based checks, such as delaying execution or logging

Implication

The imports suggest Locky creates keys, encrypts local files, and evades detection during debugging.

FLOSS and Strings Analysis

Tool: FLOSS, Strings

Findings

Obfuscation (crash after trying to parse imports table) but not found anything so we use strings to find all strings.

Implication

Since FLOSS crashes at `parseImportTable`, that's a dead giveaway the malware is **manipulating the import table**, possibly to:

- Evade static detection
- Force manual unpacking
- Hide malicious API calls

Reverse Engineering

Tool: Ghidra

Main Function

- No main function. Heavily obfuscated dynamic loading and decryption.
    - Entry point 0x004098F0 calls entry() → CRT init → **FUN_0040977a()**
    - **FUN_0040977a() → FUN_00409cc1() → &DAT_0041b5fc**
    - **FUN_0040977a() → FUN_00401360() → FUN_004050a0()**
    - void FUN_004050a0(void)
    - {
    - FUN_00407150(0x1e); → junk
    - FUN_00407330(0x1e); → junk
    - FUN_00407580(); → API loading by hashing APIs to constants
    - FUN_00407280(); → API hooking (write)
    - FUN_00406750(); → API hooking (write) + obfuscation
    - FUN_00407330(0x2a); → junk
    - FUN_00406ea0(5000); → likely an indirect call to sleep()
    - FUN_00405ce0();
        - FUN_00404770 → **RC4-like decryption** on a buffer
        - FUN_00408a20 → **Manual PE loader** into memory
        - FUN_00404100 → **Carves + splits buffers**, returns 2 structs
    - return;
    - }

Key Behaviors

- Dynamically loads and decrypts PEs, libraries, and APIs

Obfuscation

- Junk code (e.g. math problems) to complicate analysis.
- Encrypted strings decoded at runtime via RC4.
- Dynamic API calls
- Manual PE loading
- Splits buffers and structs

Pseudo-C Code (Decompiled):

```
undefined4 FUN_00405ce0(void)
{
  undefined *puVar1;
  int iVar2;
  undefined8 uVar3;
  short *psVar4;
  undefined4 local_1c;
  undefined4 local_18;

  FUN_00401a10();
  puVar1 = FUN_004053f0();
  FUN_00401420(puVar1);
  FUN_00405380();
  iVar2 = FUN_00405030();
  uVar3 = FUN_00404100(iVar2);
  local_1c = (int)uVar3;
  local_18 = (int)((ulonglong)uVar3 >> 0x20);
  *(undefined1 *)(*(int *)(local_1c + 0x10) + *(int *)(local_18 +
0x10)) = 0;
  iVar2 = FUN_00404770(*(int *)(local_1c + 4),*(int *)(local_1c +
0x10),*(uint *)(local_18 + 4),
                       *(uint *)(local_18 + 0x10));
  *(int *)(local_1c + 4) = iVar2;
  iVar2 = FUN_00404770(*(int *)(local_1c + 0x1c),*(int *)(local_1c +
0x10),
                       *(uint *)(local_18 + 0x1c),*(uint *)(local_18
+ 0x10));
  *(int *)(local_1c + 0x1c) = iVar2;
  iVar2 = FUN_00404770(*(int *)(local_1c + 0x20),*(int *)(local_1c +
0x10),
                       *(uint *)(local_18 + 0x20),*(uint *)(local_18
+ 0x10));
  *(int *)(local_1c + 0x20) = iVar2;
  FUN_00405900(*(int *)(local_1c + 0x1c),*(int *)(local_18 + 0x1c));
  FUN_00405900(*(int *)(local_1c + 0x20),*(int *)(local_18 + 0x20));
  psVar4 = *(short **)(local_1c + 4);
  puVar1 = FUN_004053f0();
  FUN_00408a20(puVar1,psVar4);
  return 0;
}
```

Implication

Locky uses sophisticated obfuscation to hide its execution process.

Dynamic Analysis (FlareVM)

Tool: ProcMon, Wireshark, and Fakenet on FlareVM Sandbox

Registry Activity

Registry writes to Internet Explorer cache settings:

- `RegSetValue HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Content`
- `RegSetValue HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Cookies`
- `RegSetValue HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\History`
- These writes manipulate browser cache paths, a behavior seen in malware attempting to **cover tracks or manipulate browsing data**.

Reads of system identity and policy settings:

- `RegQueryValue HKLM\SYSTEM\ControlSet001\Control\ComputerName\ActiveComputerName`
- `RegQueryValue HKLM\SOFTWARE\Policies\Microsoft\Windows\System`
- Common during environment fingerprinting, used for evasion or targeting.

File Activity

Access and potential modification of user profile and Temp directories:

- `CreateFile  C:\Users\admin\AppData\Local\Temp\...`
- `CreateFile  C:\Users\admin\AppData\Roaming\...`
- `CreateFile  C:\Users\admin\Desktop\...`
- These paths are commonly abused by ransomware for payload staging, encryption, and dropping ransom notes.

Execution from the Desktop:

- CreateFile C:\Users\admin\Desktop\afec2b2...exe
- Indicates the malware was executed from the Desktop, typical of user-initiated infection vectors (e.g., phishing attachments).

File type access pattern:

- Massive enumeration of files across directories, usually seen in ransomware prior to encryption.

Network Activity

No suspicious network activity

Dynamic Analysis Findings (Any.Run)

- **Tool:** Any.run Sandbox
- **Setup**: The sample was executed in an Any.Run environment (Windows 10 Professional (build: 19045, 64 bit), controlled network).

File System Activity

- No activity. Locky detects the sandbox.

Registry Activity

- Writes to
  - HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Content
  - HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Cookies
  - HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\History
- Could be hiding its tracks

Network Activity

HTTP requests	**TCP/UDP requests**	**DNS requests**	**Threats**
13	31	16	11

HTTP Requests

PID	Process	Method	HTTP Code	IP	URL
2428	afec2b2af3ace2c478382f9366f6cbc9b9579f2c9a4273150fc33a2ccd59284c.exe	POST	—	93.170.123.219:80	http://93.170.123.219/upload/_dispatch.php
			301	151.236.17.45:80	http://151.236.17.45/upload/_dispatch.php
			—	149.154.159.125:80	http://149.154.159.125/upload/_dispatch.php
			—	162.249.64.254:80	http://jtpfijqaujsdhqja.pw/upload/_dispatch.php

Connections

PID	Process	IP	Domain	ASN	CN
2428	afec2b2af3ace2c478382f9366f6cbc9b9579f2c9a4273150fc33a2ccd59284c.exe	93.170.123.219:80	—	FOP Hornostay Mykhaylo Ivanovych	RU
		151.236.17.45:80	—	M247 Ltd	DE
		151.236.17.45:443	—	M247 Ltd	DE
		149.154.159.125:80	—	M247 Ltd	DE
		162.249.64.254:80	jtpfijqaujsdhqja.pw	COMCAST-7922	US

Dns Requests

Domain	IP	Reputation
jtpfijqaujsdhqja.pw	162.249.64.254	malicious
ueqxabuscyorme.biz	—	unknown
xppkenqcd.info	—	unknown
fnowxivbmpr.info	—	unknown
pkqxpnvwprnuelpbj.info	—	unknown
vkghibslea.info	—	unknown
pclgbgn.click	—	unknown

Threats

PID	Process	Class	Message
—	—	Malware Command and Control Activity Detected	ET MALWARE Ransomware Locky CnC Beacon 21 May
—	—	Malware Command and Control Activity Detected	ET MALWARE Ransomware Locky CnC Beacon 21 May
—	—	Potentially Bad Traffic	ET DNS Query to a *.pw domain - Likely Hostile
—	—	Malware Command and Control Activity Detected	ET MALWARE Ransomware Locky CnC Beacon 21 May
—	—	Misc activity	ET INFO HTTP Request to a *.pw domain
—	—	Malware Command and Control Activity Detected	ET MALWARE Ransomware Locky CnC Beacon 21 May
—	—	Misc activity	ET INFO HTTP Request to a *.pw domain
—	—	Malware Command and Control Activity Detected	ET MALWARE Ransomware Locky CnC Beacon 21 May
—	—	Misc activity	ET INFO HTTP Request to a *.pw domain
—	—	Malware Command and Control Activity Detected	ET MALWARE Ransomware Locky CnC Beacon 21 May
—	—	Malware Command and Control Activity Detected	ET MALWARE Ransomware Locky CnC Beacon 21 May

Process Activity

Process launched itself.

Behavioral Summary

Establishes C2 communication.

Any.Run Report

- Malicious activity.
- Signatures: locky, ransomware.

Implications

Dynamic analysis reveals Locky communicates with C2 servers but detects the sandbox and stops execution.

Debugging Analysis Findings (x32 debug)

Tool: x32.dbg

API Calls Detected:

- **`ntdll.77908CC8`** - NT API calls for process management
- **`ntdll.779084E4`** - Additional NT layer functions
- Multiple **`push`** and **`call`** instructions indicating dynamic API resolution

Dynamic Loading Behavior:

- `GetInformationVirtualMemory` calls
- Memory allocation and manipulation routines
- Address space layout randomization (ASLR) bypass attempts

Memory Dump Analysis:

- ASCII strings visible: "....W....al.w"
- Partial readable text suggesting decrypted content
- Memory regions at 0x771F range (ntdll space)

RtlEnterCriticalSection Calls:

- Multiple calls to critical section management
- Thread synchronization for malware operations
- Prevents race conditions during payload execution

Bypassing Anti-Debugging Techniques

IsDebuggerPresent Check Successfully Bypassed

- The malware likely calls **`IsDebuggerPresent`** but the debugging session shows it's successfully attached
- Indicates either the check was bypassed or the malware has alternate detection methods

Incident Response Plan

Based on the NIST SP 800-61 framework, the following IR plan addresses a Locky infection in a university network:

1. Preparation

- **IR Team Formation**:
  - Define clear roles: IR Lead, Malware Analyst, Forensics Specialist, Network Admin, Communications Officer.
- **Security Infrastructure**:
  - Deploy and configure SIEM tools (e.g., Splunk) to ingest logs from endpoints, DNS, firewalls.
  - Enable host-based firewalls and enforce endpoint hardening policies.
- **User Awareness**:
  - Conduct mandatory phishing awareness training (Locky often spreads via malicious email attachments or macros).
  - Block email attachments with suspicious extensions (e.g., `.js`, `.docm`, `.exe`).
- **Backup and Recovery**:
  - Maintain isolated, immutable backups and test recovery procedures regularly.
- **Threat Intelligence Integration**:
  - Subscribe to feeds for known Locky IOCs and variants (e.g., Abuse.ch, MISP).

2. Detection and Analysis

- **Indicators of Compromise (IOCs)**:
  - MD5: **c209817538e86f5ea49fa6bd180dbf01**
  - C2 IPs: **93.170.123.219**, **151.236.17.45**, **149.154.159.125**, **162.249.64.254**
  - C2 domain: **jtpfijqaujsdhqja.pw**
- **Detection Tools**:
  - SIEM alerts on outbound HTTP POST to listed IPs/domains.
  - EDR/AV alerts for known Locky processes and DLL injection into **explorer.exe**.
  - Process monitoring shows RC4 decryption routines and manual PE loading.

3. Containment

- **Short-Term Measures**:
  - Immediately isolate infected systems from all networks.
  - Block the known malicious IPs and domains at the firewall and DNS level.
  - Disable affected user accounts and revoke suspicious access tokens.
- **Long-Term Measures**:

    - Review email filtering and disable Office macros by default via Group Policy.
    - Apply strict egress filtering and implement network segmentation to limit spread.

4. Eradication

- **Malware Removal**:
    - Delete any suspicious scheduled tasks, startup registry entries (e.g., **`HKCU\Software\Microsoft\Windows\CurrentVersion\Run\temp`**), and other persistence mechanisms introduced by Locky.
- **System Hardening**:
    - Patch vulnerable software (e.g., Windows updates, MS Office).
    - Disable unnecessary scripting engines (e.g., WScript) via group policies.
- **IOC Sweeps**:
    - Perform endpoint and network-wide sweeps using YARA or Sigma rules derived from Locky behavior.

5. Recovery

- **System Restoration**:
    - Reimage or restore systems from verified, clean backups.
    - Do *not* reconnect to the network until validated by IR and AV tools.
- **Monitoring**:
    - Enable heightened SIEM logging and EDR monitoring post-recovery for 30 days.

6. Post-Incident Activity

- **Documentation**:
    - Record infection timeline, IOCs, response actions, impact assessment, and lessons learned.
- **IR Plan Updates**:
    - Add Locky-specific behaviors and IOCs to detection rules and response checklists.
- **Staff Debriefing and Training**:
    - Conduct targeted phishing simulations based on Locky delivery techniques.
- **Security Posture Review**:
    - Review incident metrics (time to detect, time to contain) and propose improvements.

Mitigation Recommendations

- **Segmentation**: Divide network into trust zones to prevent lateral movement.
- **Threat Intel**: Automate IOC ingestion from reputable feeds like VirusTotal, Abuse.ch.
- **Email Security**: Use sandboxing for attachments, DMARC/SPF for email validation.
- **Ransom Policy**: Do not pay ransom; invest in robust backup and recovery.
- **Macro mitigation:**
    - Disable macros by default
    - Open documents in protected view
    - Use modern .docx extensions
    - Educate employees

YARA Rules

Anti-Debugging rules

1)
```
rule DebuggerCheck__API : AntiDebug DebuggerCheck {
meta:
      weight = 1
strings:
      $ ="IsDebuggerPresent"
condition:
      any of them
}
```

2)
```
rule DebuggerTiming__PerformanceCounter : AntiDebug
DebuggerTiming {
meta:
      weight = 1
strings:
      $ ="QueryPerformanceCounter"
condition:
      any of them
}
```

3)
```
rule DebuggerException__UnhandledFilter : AntiDebug
DebuggerException {
meta:
      weight = 1
strings:
      $ ="SetUnhandledExceptionFilter"
condition:
      any of them
}
```

4)
```
rule DebuggerPattern__RDTSC : AntiDebug DebuggerPattern {
meta:
      weight = 1
strings:
      $ = {0F 31}
condition:
      any of them
}
```

```
5) rule DebuggerPattern__CPUID : AntiDebug DebuggerPattern {
   meta:
        weight = 1
   strings:
        $ = {0F A2}
   condition:
        any of them
   }

6) rule DebuggerPattern__SEH_Saves : AntiDebug DebuggerPattern {
   meta:
        weight = 1
   strings:
        $ = {64 ff 35 00 00 00 00}
   condition:
        any of them
   }

7) rule DebuggerPattern__SEH_Inits : AntiDebug DebuggerPattern {
   meta:
        weight = 1
   strings:
        $a = { 64 A3 00 00 00 00 }
        $b = { 64 89 25 00 00 00 00 }
   condition:
        $a or $b
   }
```

Indicators of Compromise (IOCs)

Type	Value	Description
MD5 Hash	c209817538e86f5ea49fa6bd180dbf01	Sample hash
C2 IPs	• 93.170.123.219 • 151.236.17.45 • 149.154.159.125 • 162.249.64.254	Command and control (C2) servers
C2 Domain	jtpfiqaujsdhqja.pw	Malicious domain contacted by malware
Registry Keys	• HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Content • HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Cookies • HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\History	Modifications possibly to evade forensic analysis
URLs	• http://93.170.123.219/upload/_dispatch.php • http://151.236.17.45/upload/_dispatch.php • http://149.154.159.125/upload/_dispatch.php • http://jtpfijqaujsdhqja.pw/upload/_dispatch.php	Locky C2 beaconing and data exfiltration endpoints
Suspicious Domains	• ueqxabuscyorme.biz • xppkenqcd.info • fnowxivbmpr.info • pkqxpnvwprnuelpbj.info • vkghibslea.info • pclgbgn.click	Unresolved or low-reputation domains, potentially related

Impact of Locky Ransomware

The Locky ransomware poses a significant threat to organizational IT environments due to its sophisticated infection and evasion techniques. Its impact can be summarized as follows:

- **Data Encryption and Loss**: Locky encrypts a wide range of user and system files, rendering critical data inaccessible without the decryption key. This results in operational disruption and potential permanent data loss if backups are unavailable or compromised.

- **Operational Downtime**: The encryption of essential files leads to downtime for affected systems, impacting productivity, academic operations, and service availability in a university setting.

- **Credential Theft and Lateral Movement**: Locky's capability to steal credentials from memory and potentially leverage them to move laterally increases the scope of infection, risking further compromise of sensitive systems and data.

- **Persistence and Stealth**: The malware's use of obfuscation, API hooking, and sandbox detection allows it to persist undetected for longer periods, complicating detection and remediation efforts.

- **Network Impact**: Communication with multiple command-and-control (C2) servers creates suspicious outbound traffic, which may congest networks and increase exposure to additional payload downloads or secondary infections.

- **Forensic Evasion**: Registry and cache modifications suggest Locky attempts to hinder forensic investigations, making incident response and attribution more difficult.

- **Financial and Reputational Damage**: Beyond direct operational costs, an infection can lead to ransom demands, legal liabilities, and reputational harm, especially if sensitive academic or personal data is compromised.

- **Resource Drain**: Incident response requires significant human and technical resources, including isolating infected systems, forensic analysis, system restoration, and user credential resets.

In sum, the Locky ransomware's multi-faceted attack strategy can cause severe disruption and long-term damage, underscoring the need for comprehensive prevention, rapid detection, and efficient incident response plans.

Conclusion

This analysis of the Locky ransomware sample (MD5: **c209817538e86f5ea49fa6bd180dbf01**) reveals a sophisticated and evasive malware leveraging multiple advanced techniques to infect and persist within a Windows environment. The static analysis showed high entropy and packing, indicating strong obfuscation designed to thwart detection and analysis. Reverse engineering confirmed dynamic loading, API hashing, and runtime decryption mechanisms such as RC4, demonstrating the malware's efforts to conceal its true behavior.

Dynamic analysis in a sandbox environment revealed Locky's communication with several command and control (C2) servers, evidence of its data exfiltration attempts, and registry modifications likely intended to hinder forensic analysis. The malware's ability to detect sandbox environments and halt execution further emphasizes its stealth capabilities.

The combined use of static and dynamic provided a comprehensive view of Locky's infection lifecycle, from initial persistence mechanisms to active C2 communication. These insights enabled the formulation of a robust incident response plan aligned with the NIST SP 800-61 framework, emphasizing early detection, containment, eradication, recovery, and post-incident measures.

Given Locky's use of phishing and malicious macros for delivery, prevention strategies should focus heavily on user education, strict email filtering, and endpoint hardening. The importance of maintaining up-to-date backups and network segmentation is paramount to mitigate the risks and impacts of such ransomware infections.

In summary, this report underscores the complexity of Locky ransomware and highlights the critical need for layered defenses combining technical controls, user awareness, and rapid incident response to effectively manage ransomware threats in enterprise environments.

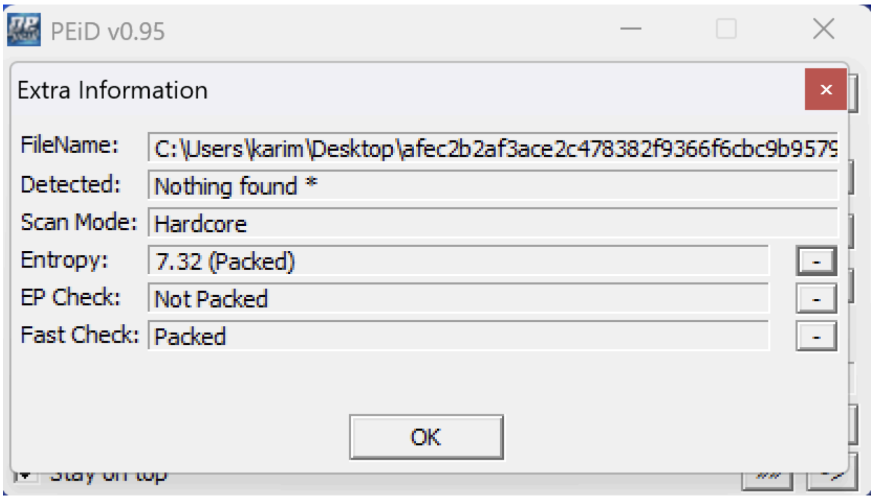
References

- Mandiant FLARE VM: https://github.com/mandiant/flare-vm
- NIST SP 800-61: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf
- MalwareBazaar: https://bazaar.abuse.ch/sample/afec2b2af3ace2c478382f9366f6cbc9b9579f2c9a4273150fc33a2ccd59284c/
- Practical Malware Analysis by Michael Sikorski and Andrew Honig.
- Any.Run : https://app.any.run/tasks/60fd4726-7466-4e3c-811c-022471f60f23

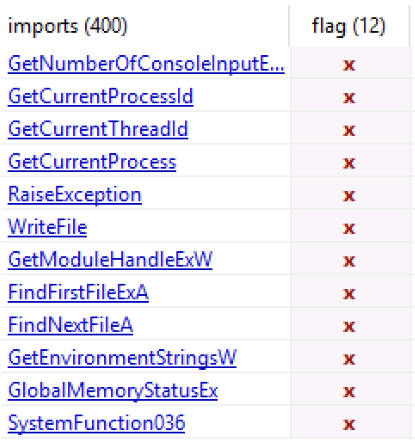
Attachments

Screenshots

PEiD output showing packing

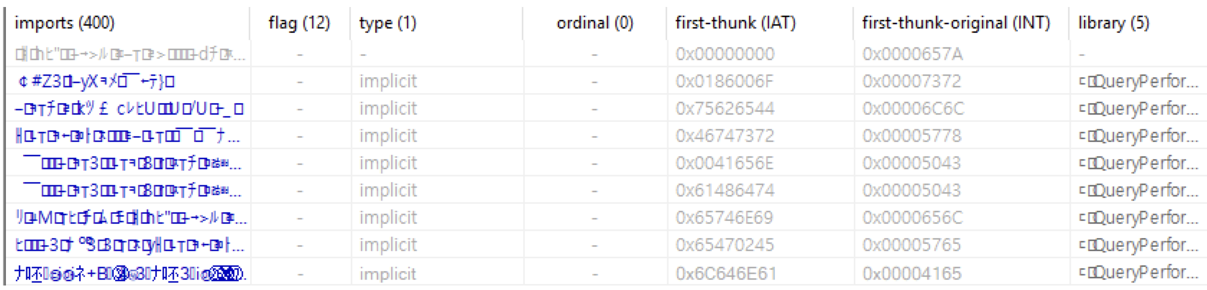


PeStudio flags imports:



imports (400)	flag (12)
GetNumberOfConsoleInputE...	x
GetCurrentProcessId	x
GetCurrentThreadId	x
GetCurrentProcess	x
RaiseException	x
WriteFile	x
GetModuleHandleExW	x
FindFirstFileExA	x
FindNextFileA	x
GetEnvironmentStringsW	x
GlobalMemoryStatusEx	x
SystemFunction036	x

PeStudio encrypted imports:



imports (400)	flag (12)	type (1)	ordinal (0)	first-thunk (IAT)	first-thunk-original (INT)	library (5)
[illegible]	-	-	-	0x00000000	0x0000657A	-
[illegible]	-	implicit	-	0x0186006F	0x00007372	QueryPerfor...
[illegible]	-	implicit	-	0x75626544	0x00006C6C	QueryPerfor...
[illegible]	-	implicit	-	0x46747372	0x00005778	QueryPerfor...
[illegible]	-	implicit	-	0x0041656E	0x00005043	QueryPerfor...
[illegible]	-	implicit	-	0x61486474	0x00005043	QueryPerfor...
[illegible]	-	implicit	-	0x65746E69	0x0000656C	QueryPerfor...
[illegible]	-	implicit	-	0x65470245	0x00005765	QueryPerfor...
[illegible]	-	implicit	-	0x6C646E61	0x00004165	QueryPerfor...

PeStudio evade imports & encrypted :

#Z3□-yX¬メ□¯←ті}□	-	implicit	-	0x46657469	0x00007373	□QueryPerfor...
□□-3□ °S□B□□□y‖□-T□←□‖□...	-	implicit	-	0x656C6946	0x00005767	□QueryPerfor...
□‖b□‖A	-	-	-	0x00000000	0x00007845	-
	-	implicit	-	0x6547023D	0x795302F1	d GetStdHand...
	-	implicit	-	0x69725074	0x6D657473	d GetStdHand...
	-	implicit	-	0x65746176	0x636E7546	d GetStdHand...
	-	implicit	-	0x666F7250	0x6E6F6974	d GetStdHand...
	-	implicit	-	0x53656C69	0x00363330	d GetStdHand...
	-	implicit	-	0x69746365	0x41564441	d GetStdHand...
	-	implicit	-	0x00416E6F	0x32334950	d GetStdHand...
	-	implicit	-	0x654702A5	0x6C6C642E	d GetStdHand...

PeStudio important strings :

unicode	31	0x000143A0	-	api-ms-win-core-datetime-l1-1-1
unicode	29	0x00013770	-	api-ms-win-core-fibers-l1-1-1
unicode	27	0x000143E0	-	api-ms-win-core-file-l2-1-1
unicode	35	0x00014418	-	api-ms-win-core-localization-l1-2-1
unicode	44	0x00014460	-	api-ms-win-core-localization-obsolete-l1-2-0
unicode	37	0x000144C0	-	api-ms-win-core-processthreads-l1-1-2
unicode	29	0x0001450C	-	api-ms-win-core-string-l1-1-0
unicode	28	0x000137AC	-	api-ms-win-core-synch-l1-2-0
unicode	30	0x00014548	-	api-ms-win-core-sysinfo-l1-2-1
unicode	28	0x00014588	-	api-ms-win-core-winrt-l1-1-0
unicode	29	0x000145C4	-	api-ms-win-core-xstate-l2-1-0
unicode	38	0x00014600	-	api-ms-win-rtcore-ntuser-window-l1-1-0
unicode	42	0x00014650	-	api-ms-win-security-systemfunctions-l1-1-0

Floss output: obfuscation

```
FLARE-VM Mon 05/05/2025 21:39:57.29
C:\Users\ANALYSIS_WIN10\Desktop>floss afec2b2af3ace2c478382f9366f6cbc9b9579f2c9a4273150fc33a2ccd59284c.exe
INFO: floss: extracting static strings
──────────────── Traceback (most recent call last) ────────────────
  in <module>:818

  in main:713

  in load_vw:418

  in getWorkspace:118

  in loadFromFile:2891

  in parseFile:38

  in loadPeIntoWorkspace:404

  in getImports:502

  in __getattr__:1498

  in parseImports:884

  in parseImportTable:1075

UnicodeDecodeError: 'utf-8' codec can't decode byte 0x90 in position 1: invalid start byte
[8032] Failed to execute script 'main' due to unhandled exception!
```

Ghidra decompiled functions showing obfuscation

Entry point

Decompile: entry - (afec2b2af3ace2c478

```
1
2 void entry(void)
3
4 {
5   ___security_init_cookie();
6   FUN_0040977a();
7   return;
8 }
```

Junk

```
2 int FUN_00407330(int param_1)
3
4 {
5   int iVar1;
6   int iVar2;
7   int iVar3;
8   int iVar4;
9   int iVar5;
10  int iVar6;
11
12  if (param_1 == 0) {
13    iVar1 = 0;
14  }
15  else if (param_1 == 1) {
16    iVar1 = 1;
17  }
18  else {
19    iVar1 = FUN_00407330(0);
20    iVar2 = FUN_00407330(param_1 + -1);
21    iVar3 = FUN_00407330(1);
22    iVar4 = FUN_00407330(0);
23    iVar5 = FUN_00407330(1);
24    iVar6 = FUN_00407330(param_1 + -2);
25    iVar1 = (iVar6 + (iVar3 - (iVar4 + 1)
26  }
27  return iVar1;
```

Obfuscation (unnecessary function call chain)

```
undefined4 FUN_00401360(void)

{
  FUN_00407330(0x28);
  FUN_004050a0();
  return 0;
}
```

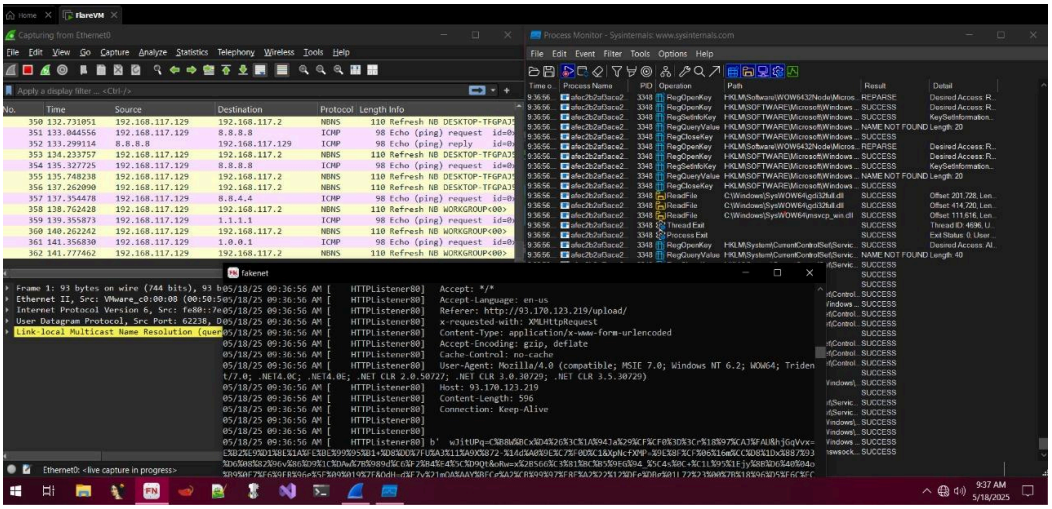
Obfuscation + junk + “Main”

```
void FUN_004050a0(void)

{
  FUN_00407150(0x1e);
  FUN_00407330(0x1e);
  FUN_00407580();
  FUN_00407280();
  FUN_00406750();
  FUN_00407330(0x2a);
  FUN_00406ea0(5000);
  FUN_00405ce0();
  return;
}
```

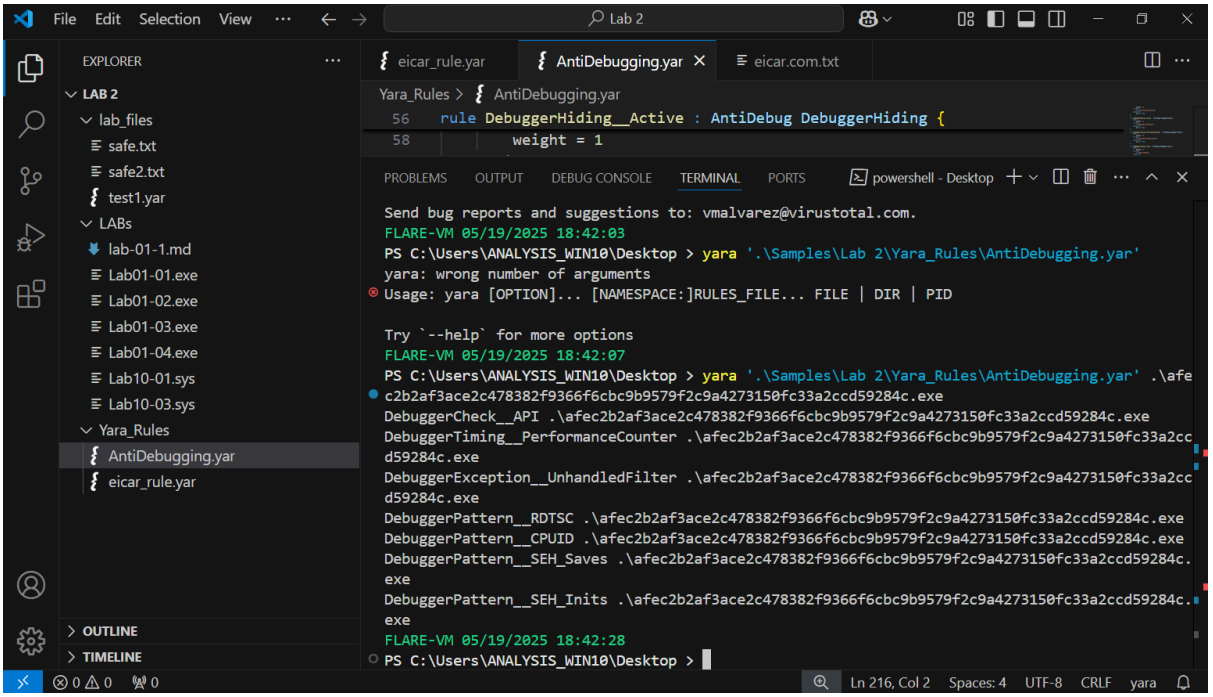
Decrypt and Load (“Main”)

```
undefined4 FUN_00405ce0(void)

{
  undefined *puVar1;
  int iVar2;
  undefined8 uVar3;
  short *psVar4;
  undefined4 local_1c;
  undefined4 local_18;

  FUN_00401a10();
  puVar1 = FUN_004053f0();
  FUN_00401420(puVar1);
  FUN_00405380();
  iVar2 = FUN_00405030();
  uVar3 = FUN_00404100(iVar2);
  local_1c = (int)uVar3;
  local_18 = (int)((ulonglong)uVar3 >> 0x20);
  *(undefined1 *)(*(int *)(local_1c + 0x10) + *(int *)(local_18 + 0x10)) =
  iVar2 = FUN_00404770(*(int *)(local_1c + 4),*(int *)(local_1c + 0x10),*(
                       *(uint *)(local_18 + 0x10));
  *(int *)(local_1c + 4) = iVar2;
  iVar2 = FUN_00404770(*(int *)(local_1c + 0x1c),*(int *)(local_1c + 0x10)
                       *(uint *)(local_18 + 0x1c),*(uint *)(local_18 + 0x1
  *(int *)(local_1c + 0x1c) = iVar2;
  iVar2 = FUN_00404770(*(int *)(local_1c + 0x20),*(int *)(local_1c + 0x10)
                       *(uint *)(local_18 + 0x20),*(uint *)(local_18 + 0x1
  *(int *)(local_1c + 0x20) = iVar2;
  FUN_00405900(*(int *)(local_1c + 0x1c),*(int *)(local_18 + 0x1c));
  FUN_00405900(*(int *)(local_1c + 0x20),*(int *)(local_18 + 0x20));
  psVar4 = *(short **)(local_1c + 4);
  puVar1 = FUN_004053f0();
```

For Dynamic Analysis
Tools : WireShark ,fakenet ,Procmon

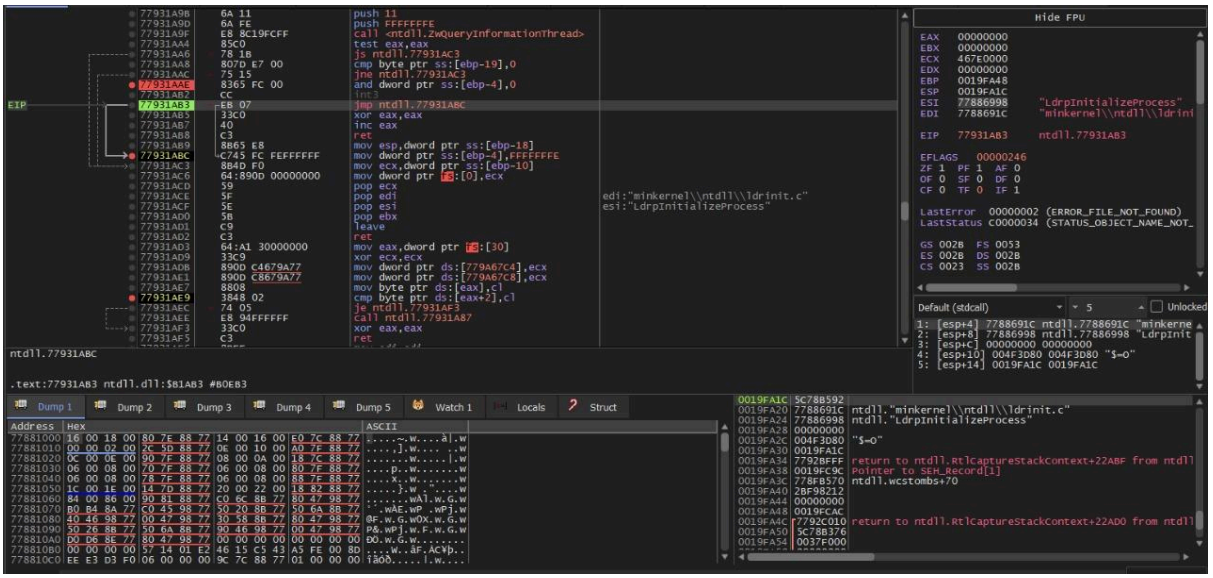


For Yara rule (AntiDebugging rules)



For Debugging (x32.dbg)

`IsDebuggerPresent`



11After bypass the debugger check