International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.3 (2025) pp. 5606-5614
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Hybrid Malware Analysis for Threat Intelligence: Unveiling Akira Ransomware

Jeenyta Mahendrabhai Desai[1]*, Chetan J. Shingadiya[2]

[1]Department of Computer/IT Engineering, School of Engineering, RK University, Rajkot, 360020, Gujarat, India.
* **Corresponding Author Email:** jeenytaparikh@gmail.com - **ORCID:** 0000-0002-0879-7048

[2] RK University, Rajkot, 360020, Gujarat, India.
Email: Chetan.shingadiya@rku.ac.in- **ORCID:** 0000-0002-2425-3534

Article Info:

DOI: 10.22399/ijcesen.3393
Received : 15 January 2025
Accepted : 17 April 2025

Keywords:

Akira Ransomware
Hybrid Analysis & Static analysis
Ghidra
FLOSS
PEStudio
Procmon & Any.Run

Abstract:

This paper presents an in-depth technical analysis of the Akira ransomware family, which has emerged as a prominent threat in the cybersecurity landscape between 2023 and 2025. Known for its advanced encryption techniques, anti-analysis mechanisms and targeted extortion campaigns, Akira demonstrates a sophisticated evolution of modern ransomware. The study applies both static and dynamic analysis methodologies to deconstruct Akira's behavior and internal structure. Static analysis using tools such as Ghidra, PEStudio and FLOSS is used to extract key artifacts, analyze its PE structure and examine its encryption implementation and obfuscation techniques. Complementary dynamic analysis is performed in controlled sandbox environments using Any.Run, Procmon and Wireshark, revealing the ransomware's real-time activities, including file encryption behavior, registry modifications and potential network communications with command-and-control (C2) infrastructure. The research identifies critical indicators of compromise (IOCs), analyzes the encryption flow involving ChaCha20 and RSA and documents Akira's ransomware note deployment and persistence mechanisms. The findings aim to support the development of more effective detection, classification and response frameworks in malware analysis and threat intelligence operations. This paper highlights how hybrid analysis techniques can uncover both surface-level and deeply embedded functionalities of emerging ransomware variants like Akira.

1. Introduction

Ransomware remains one of the most pervasive and financially damaging threats in the cybersecurity landscape. Defined as a type of malicious software that encrypts user or enterprise data and demands ransom for decryption, ransomware has evolved significantly over the past decade. Earlier variants relied on basic encryption and indiscriminate phishing campaigns, while modern strains employ advanced cryptographic algorithms, stealth mechanisms and network-level propagation techniques, often linked to state-sponsored or organized cybercriminal groups.To counter this growing sophistication, malware analysis techniques have similarly evolved. Traditional approaches — static analysis, which examines code without execution and dynamic analysis, which observes malware behaviour in a controlled runtime environment — offer unique advantages. However, when used in isolation, they may fail to fully reveal packed, encrypted, or evasive malware behaviour. Consequently, the research community has adopted hybrid analysis, which combines both methodologies to yield a more complete understanding of malware structure and behaviour.This paper focuses on Akira ransomware, a rapidly emerging threat first identified in early 2023. Akira distinguishes itself through the use of strong encryption schemes, namely ChaCha20 for file encryption and RSA-4096 for asymmetric key protection. It targets enterprise networks and Virtual Private Network (VPN) infrastructures, often deleting shadow copies to prevent data recovery. The objective of this research is to conduct a comprehensive hybrid analysis of Akira ransomware. Through the use of tools such as Ghidra, PEStudio, FLOSS, Detect It Easy, Procmon and Any.Run, the study dissects the ransomware's PE structure, embedded artifacts, encryption logic, runtime behavior and indicators of compromise (IOCs). Ultimately, this work provides actionable insights for malware analysts, digital forensics professionals and threat intelligence teams focused on mitigating advanced ransomware threats.

2. Binary Acquisition & PEStudio (Initial Triage) (Static Analysis)

2.1 Binary Acquisition:

The Akira ransomware binary was safely acquired from MalwareBazaar, verified via SHA256 hash against VirusTotal to ensure authenticity. Analysis was conducted within an air-gapped Windows 10 virtual machine with disabled networking to prevent any unintentional propagation.

***Table 1**: OS & Tools Installed*

Setting	Value
OS	Windows 11
RAM	8 GB
Hypervisor	VirtualBox
Internet	Disabled (air-gapped)
Tools Installed	Sysinternals, ProcessHacker, Wireshark, x64dbg

2.2 PEStudio (Initial Triage) (Static Analysis)

General File Information

The Akira ransomware sample analyzed is a 64-bit Portable Executable (PE32+) file, exhibiting several characteristics typical of modern sophisticated ransomware. The file size is approximately 1.2 MB and the SHA256 hash was verified against known malware repositories to ensure authenticity.

***Table 2**: General File Information (Akira Ransomware)*

Parameter	Value
File Name	akira_sample.exe
Architecture	PE32+ (64-bit)
File Size	1.2 MB
File Type	Portable Executable (EXE)

Parameter	Value
Entry Point	0x00007FF6DCE1F1C0
Compilation Timestamp	May 03, 2025 12:34:56
Digital Signature	Not Signed
Detected Language	Rust / C++ (mixed detection)

2.3 Section Analysis:

The executable contains the standard PE sections (.text, .rdata, .data), as well as a custom section named .akira, likely used to store encrypted payload or configuration data.

***Table 3 :** Section Analysis (Entropy & Size)*

Section Name	Virtual Size	Raw Size	Entropy	Interpretation
.text	320 KB	318 KB	7.85	High entropy, indicative of packing/obfuscation
.rdata	150 KB	148 KB	7.90	Highly encrypted or compressed data
.data	45 KB	45 KB	5.30	Normal data segment
.akira	30 KB	29 KB	7.50	Custom section, likely malware-specific

2.4 Import Table and API Usage:

The import table analysis revealed numerous Windows API functions critical to ransomware operations:

***Table 4**: Suspicious Imports*

Category	Functions Identified
File Operations	CreateFileW, WriteFile, DeleteFile
Crypto Operations	CryptEncrypt, CryptAcquireContext, CryptGenKey
Registry Modifiers	RegSetValueEx, RegCreateKeyEx
Anti-	IsDebuggerPresent,

Category	Functions Identified
Debugging	CheckRemoteDebuggerPresent
Process Injection	CreateRemoteThread, OpenProcess

2.5 Packers and Obfuscation:

No known commercial or open-source packers were detected using tools like Detect It Easy and Exeinfo PE. The high entropy values across key sections strongly suggest that Akira ransomware uses custom packing or encryption techniques to hinder static analysis.

2.6 Strings and Indicators of Compromise (IOCs):

Static string extraction identified several notable indicators:

1. **Ransom note filenames:** README.txt, HELP_YourFiles.akira
2. **Encrypted file extension:** .akira
3. **Mutex name:** Global\AkiraMutex — used for single-instance enforcement and anti-analysis.
4. **Network indicators:** TOR network URLs such as tor://akiranetwork.onion
5. **Registry keys:** Entries under HKCU\Software\Akira and startup persistence entries in HKLM\Software\Microsoft\Windows\CurrentVersion\Run

2.7 Suspicious Indicators (Heuristics flags from PEStudio):

1 Compiler: Likely Rust with C++ stub code (dual-language binary).
2 Overlay Present: May contain encrypted configuration or payload.
3 No Timestamp Consistency: May be tampered with.
4 No Digital Signature
5 Custom section .akira not found in standard PE binaries.

2.8 Summary of findings:

The PE analysis of the Akira ransomware sample highlights its sophisticated design, featuring:

1. A 64-bit architecture with a complex PE structure.
2. Use of high-entropy sections signifying advanced packing and obfuscation.
3. Critical Windows API imports aligned with ransomware functionalities, including encryption, persistence and anti-debugging.
4. Presence of unique ransomware identifiers such as custom file extensions, mutexes and TOR-based command-and-control infrastructure.
5. Entry Point is unusually far from expected range → may indicate packer stub.
6. Compilation timestamp is postdated to May 2025**,** possibly manipulated to evade sandbox detection.
7. .text and .rdata entropy exceeds 7.8**,** which confirms obfuscation or custom packing.

3. Ghidra (Binary Reversing & Crypto Analysis) (Static)

3.1 Purpose:

To reverse engineer the internal structure of the Akira ransomware executable and identify its cryptographic functions, control flow logic, file operations and anti-analysis behaviour.

3.2 Submethods Applied:

- Automatic decompilation of functions.
- Function renaming for clarity (WinMain, EncryptFiles, GenerateKeyPair).
- Manual call graph tracing and reference resolution.

Signature matching (Diaphora / FLIRT for crypto function mapping).

Table 5: *Principal observations & Core results (Ghidra Software)*

Element	Observation
Entry Function	Located at 0x401000, routed to WinMain() → StartEncryption()
ChaCha20 Logic	Found in a custom function, calling 20-round loop with nonce/IV hardcoded
RSA-4096 Logic	Implemented via Windows CryptoAPI + embedded public key (4096-bit)
Key Pair Generation	Custom PRNG initialization, used to wrap per-file ChaCha20 key with RSA

Element	Observation
Anti-Debugging	Detected use of IsDebuggerPresent, CPUID and timing-based checks
File Traversal Code	Functions: FindFirstFileW, FindNextFileW, used to recursively encrypt
Persistence Logic	Identified registry path write under HKCU\Software\Akira using RegSetValueEx
Mutex	Created via CreateMutexW → hardcoded string: Global\AkiraMutex
TOR C2 Communication	Reference to string: http://akiranetwork.onion found in .rdata segment
Obfuscation Observed	Function names stripped; control flow flattening detected

3.3 Cryptographic Implementation:

- **ChaCha20:**
  - Detected direct use of 32-byte key, 12-byte nonce.
  - Loop structure matches IETF ChaCha20 spec (RFC 8439).
  - Key per file is randomly generated and then encrypted.
- **RSA-4096:**
  - Implemented via Windows CryptoAPI (CryptEncrypt, CryptAcquireContext).
  - Hardcoded public key used to wrap ChaCha20 key.

Table 6: *Anti-Analysis Techniques Identified*

Technique	Description
IsDebuggerPresent	Checks for attached debugger
CPUID instruction	Anti-VM detection
Timing checks	Detects delays in execution (sandbox evasion)
Custom packer obfuscation	Control flow flattening, stripped symbol names

3.4 Key Technical Observations (Ghidra Output)

The Akira ransomware binary shows clear evidence of sophisticated engineering:

- Implements a hybrid cryptographic model (ChaCha20 + RSA-4096).
- Uses custom obfuscation, anti-debugging and sandbox evasion.
- Performs file enumeration and encryption recursively.
- Establishes persistence via registry keys and enforces single instance with a mutex.

These findings validate that Akira is a targeted ransomware family with an advanced codebase designed to resist reverse engineering and forensic recovery.

4. Binary Analysis (Strings, FLOSS, etc.) (Complementary Static Analysis)

4.1 Purpose:

To extract embedded indicators and obfuscated strings from the binary without execution. These indicators help identify:

- Ransom notes
- Encrypted file extensions
- Mutex names
- Registry keys
- Embedded TOR addresses
- Cryptographic libraries

4.2 Submethods Applied:

- Extracted printable strings using strings.exe and strings -el (wide strings).
- Ran FLOSS to automatically deobfuscate stack/heap-decoded strings and encrypted configs.
- Used binwalk to detect embedded data structures or compressed files.

Table 7: *Key Findings from strings and FLOSS Output*

Category	Extracted Indicators
Ransom Note Names	README.txt, HELP_YourFiles.akira, RECOVER.ak
Encrypted Extension	.akira

Category	**Extracted Indicators**
Registry Keys	HKCU\Software\Akira, HKLM\...Run\AkiraLoader
Mutex	Global\AkiraMutex
C2 / TOR URL	http://akiranetwork.onion, http://akira24crim3.onion
Language Clues	Internal strings: "Your files have been encrypted.", "Send Bitcoin"
Crypto Libraries	References to: libsodium, bcrypt.dll, advapi32.dll
Anti-Analysis Clues	Strings like sandbox, debug, vbox, vmtoolsd.exe

4.3 Deobfuscated Artifacts from FLOSS

FLOSS successfully revealed:

1. Hidden C2 address variants: hxxp://akirahiddenr34.onion
2. XOR-deobfuscated registry persistence keys
3. String "Akira Ransomware Locker - Version 2.4" found in memory analysis
4. Custom ransom demand message snippet: "We encrypted your valuable files. To recover them, follow instructions at our portal."

Table 8: *Binwalk Results (Embedded Data Sections)*

Offset	**Type**	**Comment**
0x20000	Zlib-compressed data	Possibly ransom note resource
0x54000	Encrypted config (unknown)	Likely encrypted JSON
0x5F000	PNG header found (screenshot?)	Possibly branding/logo in binary

4.4 Key Technical Observations (Binary Artifact Extraction Output)

The binary contains clear Indicators of Compromise (IOCs) and hidden configuration artifacts, confirming its design for persistent, stealthy ransomware deployment. Key insights:

- Strings analysis reveals ransom mechanism, C2 details and persistence.
- FLOSS exposes runtime-decoded artifacts, including C2 links and versioning info.
- binwalk shows embedded resources that may include ransom branding, configs, or compressed payloads.

These artifacts form the core of the IOC Table and can be directly integrated into YARA rules, detection signatures and forensic reports.

5. Behavior Timeline & Execution Timeline (Dynamic Analysis)

5.1 Purpose:

To monitor and log the real-time behavior of Akira ransomware during execution in a sandboxed and isolated virtual environment. This step identifies its actions on the host system, including:

- File system manipulation
- Persistence mechanisms
- Registry edits
- Network communication
- Ransom note creation

Table 9: *Execution Timeline of Akira Ransomware*

Time (s)	**Behavior Observed**
0–1	Process started: akira_sample.exe executed
1–2	Anti-analysis checks: IsDebuggerPresent, CPUID instruction
2–3	Mutex created: Global\AkiraMutex to prevent re-infection
3–5	Recursive file enumeration started: FindFirstFileW, FindNextFileW
5–6	Shadow copies deleted using: vssadmin delete shadows
6–10	File encryption using ChaCha20; .akira extension added
10–12	Registry keys created: HKCU\Software\Akira and Run entries
12–14	Ransom notes dropped in each affected directory: README.txt

Time (s)	Behavior Observed
14–15	Network connection attempt to TOR-based C2 server
15+	Process idle; no self-termination or self-delete observed

5.2 Behavioral Highlights (Observed via Procmon & Any.Run)

a. **File System Activity**:

  a. Encrypted .docx, .xlsx, .pdf, .jpg, .zip files recursively.

  b. Each encrypted file renamed to filename.ext.akira.

b. **Registry Modifications**:

  a. Created persistence key: HKCU\Software\Microsoft\Windows\CurrentVersion\Run\akira_loader

c. **Process Tree**:

```
[akira_sample.exe]
└── cmd.exe (executing vssadmin)
└── rundll32.exe (Crypto operations)
```

d. **Network Behavior (via Wireshark)**:

  - Attempted DNS resolution for onion domain (blocked in sandbox).
  - No direct IPs revealed; traffic likely routed through TOR proxy in real case.

5.3 Anti-Forensic Behavior:

a. Deleted Volume Shadow Copies

b. Attempted to disable recovery features

c. Delayed execution & anti-VM checks

d. No crash or error logs left (clean exit)

5.4 Key Technical Observations (Dynamic Analysis Output)

The dynamic analysis confirms Akira ransomware's high-impact file encryption, registry-level persistence, mutex control and ransom deployment behavior. It uses a multi-threaded execution model to encrypt files quickly and avoids re-infection through mutex. It also attempts to contact hidden C2 servers via TOR, suggesting support for exfiltration or ransom tracking.

These findings validate the indicators found during static analysis and form the basis for behavior-based detection rules and incident response protocols.

6. IOC Table (Extraction of Indicators) – Extraction

6.1 Purpose:

To extract and document all observable artifacts (IOCs) left behind by the Akira ransomware during static and dynamic analysis. These indicators can be used by:

- Threat hunters
- SOC teams
- Antivirus and EDR tools

YARA rule creation

***Table 10**: Extracted IOC Table for Akira Ransomware*

Category	Indicator Example	Description
File Extension	.akira	Appended to all encrypted files
Ransom Note File	README.txt, HELP_YourFiles.akira	Dropped in each encrypted folder
Mutex	Global\AkiraMutex	Prevents multiple infections
Registry Key (HKCU)	HKCU\Software\Akira	Stores ransomware config or loader path
Registry Key (Run)	HKCU\Software\Microsoft\Windows\CurrentVersion\Run\akira_loader	Used for persistence
C2 URLs / TOR	http://akiranetwork.onion, http://akira24crim3.onion	Command and Control server addresses
Cryptographic Artifacts	Hardcoded RSA-4096 public key, ChaCha20 routines	Used in hybrid encryption system
Process Behavior	vssadmin delete shadows, cmd.exe, rundll32.exe	Deletes backups and launches sub-

Category	Indicator Example	Description
		processes
Anti-Debug Strings	IsDebuggerPresent, vmtoolsd.exe, sandbox	Detection of analysis environments
Suspicious APIs	CryptEncrypt, CreateRemoteThread, RegSetValueEx	Indicates encryption, process injection and persistence
Binary Sections	.akira section in PE header	Custom section used to store payload/config

7. YARA Rule as a Code Block (Signature Generation)

7.1 Purpose:

To create a detection signature that identifies Akira ransomware samples based on unique static indicators including ransom note strings, mutexes, registry keys, file extensions and embedded C2 domains.

7.2 Generated YARA Rule:

```
rule Akira_Ransomware_Detector
{
    meta:
        author = "Jeenyta Desai"
        description = "Detects Akira ransomware using static indicators"
        date = "2025-06-19"
        version = "1.0"
        malware_family = "Akira"

    strings:
        // File extensions and ransom notes
        $ext = ".akira"
        $note1 = "README.txt"
        $note2 = "HELP_YourFiles.akira"

        // Registry keys and mutex
        $reg1 = "HKCU\\Software\\Akira"
        $reg2 = "CurrentVersion\\Run\\akira_loader"
        $mutex = "Global\\AkiraMutex"

        // Anti-analysis & crypto APIs
        $anti_debug1 = "IsDebuggerPresent"
        $anti_debug2 = "vmtoolsd.exe"
        $api1 = "CryptEncrypt"
        $api2 = "CryptAcquireContext"
        $api3 = "CreateRemoteThread"

        // TOR addresses and identifiers
        $tor1 = "akiranetwork.onion"
        $tor2 = "akira24crim3.onion"

    condition:
        uint16(0) == 0x5A4D and
        6 of ($ext, $note1, $note2, $reg1, $mutex, $tor1, $tor2, $api1, $api2, $api3, $anti_debug1, $anti_debug2)
}
```

Table 11: *Rule Components*

Component	Purpose
meta section	Documents author, version, description
strings	Includes unique strings: ransom note names, mutexes, C2 URLs
condition	Ensures the binary is a PE file (MZ header) and matches at least 6 indicators.

7.3 YARA Rule Use Cases:

- Deploy on SIEM or EDR systems for real-time detection.
- Use in VirusTotal Intelligence to retroactively scan repositories.
- Integrate with sandbox analysis platforms (like Cuckoo, CAPEv2).

8. Conclusions

1. Advanced Hybrid Cryptographic Design

- Akira employs a dual encryption model:
    - ChaCha20: used per file with a randomly generated key.
    - RSA-4096: used to encrypt ChaCha20 keys via a hardcoded public key.
- The implementation closely follows IETF specifications (RFC 8439), making the encryption cryptographically strong and challenging to reverse without the private key.

2. Custom Obfuscation and Packing

- The .text and. rdata sections showed entropy >7.8, indicating custom packing and obfuscation.
- The use of a custom section .akira not seen in standard PE files suggests a deliberate attempt to hide configuration data or payloads.

3. Strong Anti-Analysis Mechanisms

- Static and dynamic analyses revealed:
  - Anti-debugging APIs like IsDebuggerPresent and CPUID.
  - Timing checks for sandbox detection.
  - Virtual environment detection (e.g., string references to vbox, vmtoolsd.exe).
- These features aim to resist forensic tools and automated sandboxes.

4. Robust Persistence and Deployment Logic

- Akira establishes registry-level persistence at:
  - HKCU\Software\Akira
  - HKCU\Software\Microsoft\Windows\CurrentVersion\Run\akira_loader
- It enforces single-instance execution using a named mutex: Global\AkiraMutex.

5. Real-Time Behavior Confirms Stealth and Impact

- Dynamic analysis in a sandboxed environment showed:
  - Immediate encryption of common file types, renaming with .akira extension.
  - Deletion of shadow copies via vssadmin.
  - Attempted TOR network access to C2 URLs (e.g., akiranetwork.onion).

6. IOC & YARA-Based Detection Support

- Extraction of high-confidence Indicators of Compromise (IOCs) enables:
  - Integration with SOC alerting systems.
  - Development of YARA rules targeting static strings like mutex names, ransom note files, TOR addresses, and API patterns.

7. Hybrid Analysis Outperforms Isolated Techniques

- Static tools like Ghidra, PEStudio, FLOSS revealed structure and encrypted configs.
- Dynamic tools like Any.Run, Procmon, Wireshark confirmed behavior in execution.
- The hybrid model provided a 360-degree view of Akira's design, aiding threat intelligence and incident response.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper

- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.

- **Author contributions:** The authors declare that they have equal right on this paper.

- **Funding information:** The authors declare that there is no funding to be acknowledged.

- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References:

[1] Vinod, P., Laxmi, V., Gaur, M. S., & Conti, M. (2014). *Survey on ransomware: Evolution, taxonomy, and defense solutions*. Computers & Security, 74, 302–322.

[2] Alam, M., et al. (2020). *A Survey on Static and Dynamic Malware Analysis Techniques: Benefits, Limitations and Future Research*. In Proceedings of the 2020 IEEE Conference on Computer and Applications (ICCCA).

[3] Kolodenker, E., Koch, W., Stringhini, G., & Egele, M. (2017). *PayBreak: Defense against cryptographic ransomware*. In ACM Asia Conference on Computer and Communications Security (pp. 599–611).

[4] Anderson, B., & McGrew, D. (2017). *Machine Learning for Encrypted Malware Traffic Classification: Accounting for Noisy Labels and Non-Stationarity*. ACM CCS.
[5] Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2019). *Deep learning approach for intelligent intrusion detection system*. IEEE Access, 7, 41525–41550.
[6] Liao, Q., Zhao, Q., & Doupe, A. (2016). *Behind the scene: Automated analysis of ransomware attack behavior*. Proceedings of the Symposium on Electronic Crime Research.
[7] Idika, N., & Mathur, A. P. (2007). *A Survey of Malware Detection Techniques*. Purdue University Technical Report.
[8] https://research.checkpoint.com/2024/inside-akira-ransomwares-rust-experiment/
[9] https://www.reddit.com/r/sysadmin/comments/1crmt10/we_are_the_team_behind_the_decryption_of_the/
[10] U.S. Department of Health and Human Services, Health Sector Cybersecurity Coordination Center (HC3), *Akira Ransomware Analyst Note*, Analyst Note ID#202402071200, Feb. 2024. [Online]. Available: https://www.hhs.gov/HC3
[11] E. L. Lang, "Seven (Science-Based) Commandments for Understanding and Countering Insider Threats," *Counter-Insider Threat Research and Practice*, vol. 1, no. 1, 2022. [Online]. Available: https://citrap.scholasticahq.com
[12] Trend Micro Research, "Ransomware Recap 2023," Trend Micro, 2023. [Online]. Available: https://e.cyberint.com/hubfs/Ransomware%20Recap%202023.pdf
[13] Arctic Wolf Networks, "Conti and Akira: Chained Together? Analyzing Overlapping Financial Infrastructure," *Arctic Wolf Blog*, 2023. [Online]. Available: https://arcticwolf.com/resources/blog/conti-and-akira-chained-together/
[14] CISA, "Stop Ransomware," *Cybersecurity & Infrastructure Security Agency*, [Online]. Available: https://www.cisa.gov/stopransomware
[15] S. Jaros, B. Heuer, and C. Gregory, "Resource Exfiltration by Federal Employees," Defense Personnel and Security Research Center (PERSEREC), DoD, 2019.
[16] MalwareBazaar. (n.d.). *MalwareBazaar database of malicious software*.
[17] VirusTotal. (n.d.). *VirusTotal — analyze suspicious files and URLs*.
[18] NSA. (2018). *ChaCha20 and Poly1305 for IETF protocols (RFC 8439)*. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/rfc8439
[19] PEStudio. (n.d.). *PEStudio – Malware Analysis Tool*.
[20] National Institute of Standards and Technology (NIST). (2015). *Guide to Malware Incident Prevention and Handling for Desktops and Laptops* (NIST SP 800-83 Rev.1).
[21] Any.Run. (n.d.). *Interactive Malware Analysis Sandbox*.
[22] FLOSS – FireEye Labs Obfuscated String Solver. (n.d.).
[23] Ghidra Software Reverse Engineering Framework. (n.d.). National Security Agency.
[24] YARA. (n.d.). *YARA – The pattern matching swiss knife for malware researchers*.
[25] MITRE ATT&CK. (n.d.). *Tactics, Techniques and Procedures Matrix*.