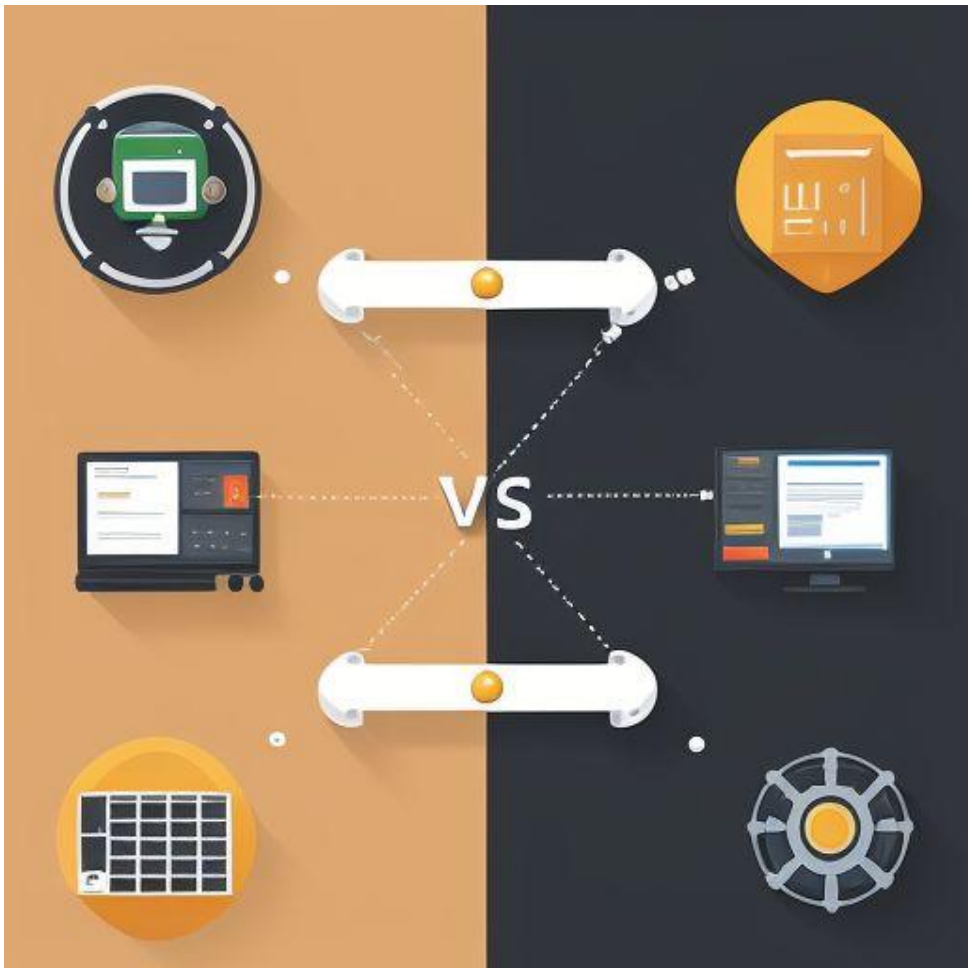
Detecting Malicious Communication in Air-Gap Systems - Software Based Detection System

Kelly Zhang, Nahit Söğütlü



Index

Context

[Nahit]Air-gap systems, designed to be isolated from external networks, are widely regarded as one of the most secure configurations for protecting sensitive data and critical infrastructure. Physical separation from the internet or other networks significantly reduces the risk of direct cyberattacks. However, this isolation does not guarantee absolute security, as sophisticated adversaries have increasingly found ways to exploit vulnerabilities through indirect communication channels. Malicious communication, often hidden or disguised within normal system operations, poses a growing threat to air-gap systems, making traditional security measures insufficient.

Problem: Hardware-based detection systems are hard to implement, costly and not always reliable to secure air-gapped systems.

First, we analyze different types of malware targeting air-gapped systems. For each case, we document the characteristics, specifically how the malware collects and transmits data. This helps us understand which techniques are commonly used.

Based on these characteristics, we can later define criteria for designing and building an effective detection system and overall security design.

Hardware-based defense systems are sometimes considered to protect air-gap systems. However, these methods come with significant challenges. Techniques such as jamming can disrupt both legitimate and malicious communications, requiring high-powered equipment that may not always be effective against advanced attacks. Moreover, a noise cancellation technique can be unreliable, often unable to distinguish between benign and malicious signals. Frequency analysis, while useful, is costly and may struggle against modern techniques like frequency hopping or encryption. These hardware-based defenses are often impractical as standalone solutions, highlighting the need for a more reliable approach.

[Kelly]This paper introduces a Software-Based Detection System designed to identify and prevent malicious communication in air-gap environments. By focusing on the detection of unauthorized data transfer attempts and covert channels, this system leverages software algorithms to analyze abnormal system behaviors. The proposed approach offers a proactive defense mechanism, enhancing the resilience of air-gap systems against evolving threats and ensuring that even the most covert forms of communication are swiftly identified and neutralized. The paper tries to answer these questions:

- What kind of software-based detection systems are there?

- What kind of software-based detection systems are useful?
- How should a software-based detection system for air-gapped systems be?
- Comparison of software and hardware-based systems.
- How does air-gapped malware operate? What are the common behaviors?

Analysis

[Kelly]In this section, we will first analyze existing malware. We have divided these into two categories:

1. **Previously Detected Malware in Air-Gap Systems**
   These are real-world examples of malware that have been discovered and documented in air-gapped environments.

2. **Experimental Malware Examples**
   These are malware concepts or prototypes developed and described in academic research papers.

For each malware example, we will describe its characteristics and explain what it does on the air-gapped machine.

At the end of the section, we will provide an analysis of different types of defense systems. For each defensive element, we will discuss the pros and cons in the context of protecting air-gapped systems.

Previously Detected Malware in Air-Gap Systems

[Kelly]USBStealer

- USBStealer automatically exfiltrates collected files via removable media when an infected device connects to an air-gapped victim machine after initially being connected to an internet-enabled victim machine.[22]
- USBStealer drops commands for a second victim onto a removable media drive inserted into the first victim, and commands are executed when the drive is inserted into the second victim. [22]
- USBStealer searches victim drives for files matching certain extensions (".skr",".pkr" or ".key") or names. [22], [23]
- USBStealer has several commands to delete files associated with the malware from the victim.[22]
- USBStealer sets the timestamps of its dropper files to the last-access and last-write timestamps of a standard Windows library chosen on the system. [22]
- USBStealer monitors victims for insertion of removable drives. When dropped onto a second victim, it also enumerates drives connected to the system.[22]

[Nahit]Ramsay

- Ramsay can conduct an initial scan for Microsoft Word documents on the local system, removable media, and connected network drives, before tagging and collecting them. It can continue tagging documents to collect with follow up scans. [24]
- Ramsay can collect Microsoft Word documents from the target's file system, as well as .txt, .doc, and .xls files from the Internet Explorer cache. [24]
- Ramsay can take screenshots every 30 seconds as well as when an external removable storage device is connected. [24]
- Scans for removable drives [25]

Malware cannot jump air gaps without some novel functionality that is as of yet unknown to the general information security public. This does not mean that air-gapped computers can't become infected; when they do, it is normally due to infected removable drives. Researchers currently find that the most likely way for Ramsay to jump the air gap is by infecting PE on removable drives, where the malware is downloaded when the file is executed. [25]

[Nahit]Usbferry

- USBferry can check for connected USB devices. [26]
- USBferry can copy its installer to attached USB storage devices. [26]
- USBferry can check for connected USB devices. [27]
-

By using "tracert" and "ping" commands, the group discovers the target's network architecture. For example, "tracert -h 8 8.8.8.8" collects the route (path) and measures transit delays of packets across an Internet Protocol (IP) network. The pings, meanwhile, are used to test the target network's connectivity; the actors want to know if the machine has access to the internal network and the target mail portal. Tropic Trooper can then use the gathered information to plan and prepare the next stage of the attack. [27]

[Kelly]USBCulprit

- The brunt of an attack from USBCulprit is that the malware is capable of scanning multiple paths on an infected system. It looks for the following: .doc, .docx, .pdf, .xls, .xlsx, .ppt, .pptx, .wps, .rtf .20]
- compromised machine as a side-loaded DLL which is implanted onto signed, legitimate applications [20][21].

- RedCore records the title of the current foreground window (if it exists) and logs keystrokes each 10ms to an internal buffer of size 65530. When this buffer is filled, data from it is written to a file named 'RCoRes64.dat'. The data is encoded using a single byte XOR with the key 0xFA. (keylogger) [21].
- USBCulprit hides file extensions (and ensures the user cannot view hidden files) by modifying registry keys with its CUSB::RegHideExt and CUSB::RegHideFile functions [20].
- scans the compromised system to find files to steal by scanning for the list of file extensions listed above with its CUSB::USBFindFile function, then grouped into RAR archives and encrypted [20].
- If it detects that the drive is a USB, the malware then decides whether to copy those RAR archives or to use the USB drive as another source of information to steal. It then searches the drive for a directory called $Recyc1e.Bin. If it does not yet exist, it will create the directory. This becomes the target path for the stolen information to be stored in on the USB drive. [20]

[Kelly]Agent.btz

- Agent.btz drops itself onto removable media devices and creates an autorun.inf file with an instruction to run that file. When the device is inserted into another system, it opens autorun.inf and loads the malware. [28]
- Agent.btz creates a file named thumb.dd on all USB flash drives connected to the victim. This file contains information about the infected system and activity logs. [28]
- Agent.btz saves system information into an XML file that is then XOR-encoded. [28]
- Agent.btz obtains the victim username and saves it to a file. [28]
- Agent.btz creates a file named thumb.dd on all USB flash drives connected to the victim. This file contains information about the infected system and activity logs. [29]

[Kelly]Fanny

- USB Worm with a sophisticated backdoor that uses the so-called "Stuxnet LNK vulnerability" to automatically execute from the USB drive even if Autorun has been disabled.[30]
- it uses the USB drive as a carrier to send/receive requests to and from the operator via a hidden storage area created in raw FAT structure [30]

- It copies a .dll file to the system and will remain the value of remained maximum length of the Fanny kill chain in the removable drive. It also copies all *.lnk files from the USB drive to “%WINDIR%\system32\” in order to reuse them when infecting other attached USB drives. Note that there may be more than one LNK file, because each LNK contains a distinct path to the DLL which gets loaded. [30]
- Detect is there are existing mutexes, so that the same code won't run twice [30]

Note : When the counter reaches a minimal value of one the module cleans up the USB drive and stops spreading the worm.

[Nahit]SysmonLoader

- Targets older machines
- Not part of an ongoing campaign
- Infection though compromised USB drives, the how is unknown

“Specifically, we do not know if there has been a successful compromise in the supply-chain making these devices, or if these have been compromised post-manufacturing and distributed using other means such as social engineering.”

“The attacker encrypted the unknown executable file and concealed it at the ending part of the secure USB storage in advance. The hidden data is not accessible through logical file operation APIs, such as ReadFile(). Instead, SymonLoader uses Logical Block Addressing (LBA) and SCSI commands to read the data physically from the particular expected location on the removable drive,” the researchers explain.[32]

[Kelly]Brutal Kangaroo

- USB drive infects with the help of a flaw in the Microsoft Windows operating system that can be exploited by hand-crafted link files (.lnk) to load and execute programs (DLLs) without user interaction.[31]
- Brutal Kangaroo components create a custom covert network within the target closed network and providing functionality for executing surveys, directory listings, and arbitrary executables.[31]
- Once multiple Shadow instances are installed and share drives, tasking and payloads can be sent back-and-forth," the manual reads.[31]
- new commands into an infected air-gapped network after the initial Shadow tasks have been executed and initial data has been retrieved (via Broken Promise installed on the primary host).Initial and additional Shadow tasks can be configured via a builder application, similar to the one used by Drifting Deadline.[31]

[Kelly]Flame

- Flame contains modules to infect USB sticks and spread laterally to other Windows systems the stick is plugged into using Autorun functionality. [33]
- Flame can collect AutoCAD design data and visio diagrams as well as other documents that may contain operational information. [34]
- Flame has a module named BeetleJuice that contains Bluetooth functionality that may be used in different ways, including transmitting encoded information from the infected system over the Bluetooth protocol, acting as a Bluetooth beacon, and identifying other Bluetooth devices in the vicinity. [35]

Experimental Malware Examples

These malware examples are experimental and have not been detected in any real

Electromagnetic

[both]Rambo

[kelly]The Rambo attack is based on data that moves through the RAM bus, it causes rapid voltage and current changes that create electromagnetic emissions. These emissions can be modulated to represent binary data. So, in short when sending 1's you send data to go through the RAM bus and 0's you do nothing.

When going through the RAM bus you first need to bypass the cache this is done by instructions like MOVENTI, STR. There are also other methods by first filling the cache, then you can also send data through the RAM bus reliably.

For measuring it is important to find instructions that can bypass the cache (MOVENTI etc.) and programs/ functions that are done repeatedly which are used for filling the cache and going through the RAM bus.

(Nahit's Note about the MOVENTI and STR instructions: Initially we all went through the papers and understood the purpose of these instructions. I believe I came up with the idea of statically checking for these instructions for the defense side. Since these instructions (called non-temporal access) are uncommon among legitimate software. The reason is that their legitimate use case is for optimization, and not every program needs them. I tested this by reverse engineering several programs such as Firefox, Audacity, OBS, and sqlitebrowser. Therefore, they can be an indicate (not conclude) that the virus is using a Rambo-like attack.)

Magnetic

[Kelly]Odini

ODINI uses low-frequency magnetic fields emitted by the CPU when it is under varying workloads. These magnetic fields can be modulated (manipulated in patterns) to encode binary data (1s and 0s). For receiving the magnetic fields, it uses a magnetometer.

For measuring we should focus on tracing activities on the system and find repeatedly “on” and “off” workloads and overloads in CPU cores. Moeover, we could also find script that abnormally regulates the workload.

Optical

[Kelly]PrinterLeak

Here they use malware to find a printer in an airgap system, then data will be exfiltrated from the printer by led modulation(blinking), lcd screen context by displaying QR code or text and lcd screen brightness where only the background color is modulated but the content is still the same. For retrieving the data this project uses a camera filming the printer.

Malware for printer control / display panel:

- firmware attack:
    - low lever registers / GPIO
    - C&C with printer via protocols (http request, telnet, smb etc.) needs high degree of control over display!

Remote configuration:

- PJL (printer job language), PJL command %-12345X@PJL RDYMSG DISPLAY= "SECRET PASSWORD"%-12345X
- (modern) using http reqeust to configurate commands

Remote operation attack:

- uses control commands
- PCL (printer control language), PLJ, IPP (internet printing protocol), SMB, etc.
- http commands or dedicated ports (9100(appsocket), 515(line printer daemon), 631(internet printing protocol))

Acoustic

[Nahit]Pixhell

Explanation

The **PIXHELL attack** is a sneaky way hackers can steal information from computers that aren't even connected to the internet (air-gapped). It works by making **LCD screens give off tiny sounds,** not from speakers, but from the screen's own electronics, when certain images (pixel patterns) are shown.

A malware on the computer can control these patterns to send out **hidden sound signals** that carry data, like passwords. These sounds are too quiet for people to hear but can be picked up by nearby devices like phones. It can send small bits of info from up to **2 meters away**, even if the screen looks black.

Countermeasures

Anomaly Detection Systems. Implementing anomaly detection systems that identify suspicious behavior on the bitmap shown on the screen. In the case of a PIXELL attack, an analysis of the screen buffer can be used to detect the transmission pattern, which usually consists of white pixels on iterative patterns that encode a specific frequency. Such a solution can be implemented at the user or kernel level. For example, utilizing GDI functions to capture the screen. In Windows OS, Win32 API functions such as BitBlt() [36] *can be used to copy the screen buffer or a window into a bitmap. Such security solutions can be bypassed by malware running on a compromised computer at the user and kernel levels. [36]*

[Nahit]Hardware Based Defense for Air-Gap Systems

Hardware Based Systems

Hardware-based defense mechanisms for air-gapped systems focus on physical isolation and signal disruption to prevent unauthorized data exfiltration. These systems often employ electromagnetic shielding, data diodes, and physical disconnection of network interfaces to enforce strict isolation [2][3]. Techniques like jamming covert radio frequencies (RF) or acoustic channels aim to disrupt malicious communication attempts, while unidirectional network gateways (data diodes) allow controlled one-way data flow to prevent reverse infiltration.

Common hardware approaches include:

- **Electromagnetic shielding** to block unintended RF emissions from components like USB connectors or internal buses that could transmit data via GSM frequencies
- **Acoustic dampening** to neutralize malware using inaudible sound waves (e.g., AirHopper's FM-based exfiltration) [2][4]
- **Thermal monitoring systems** to detect BitWhisper-style heat fluctuations used for bidirectional communication [4][5]
- **Infrared blockers** to prevent bridge attacks via compromised surveillance cameras [2]

While effective against basic threats, hardware solutions face inherent limitations: Common hardware-based defense systems such as jamming, noise cancellation and frequency analysis tools often prove cost-prohibitive and struggle with advanced malware such as NoiseHopper and RAMBO that use advanced techniques such as frequency-hopping and DDR emanations [6][7]. Data diodes, though reliable for unidirectional transfer, cannot prevent malware from establishing outbound communication through alternative channels like power line fluctuations or optical signals

[Kelly]Software Based Defense for Air-Gap Systems

In contrast to hardware-based methods, software-based detection systems offer a lightweight and cost-effective means of identifying threats within air-gapped environments. These systems are deployed directly on the host and monitor for anomalies or malicious activities by analyzing file integrity, application behavior, process spawning patterns, and system resource usage. While such systems cannot prevent all covert channel communication, they excel at detecting post-exploitation behavior, such as malware attempting to use non-standard techniques for data exfiltration.

File Integrity Monitoring (FIM)

File Integrity Monitoring (FIM) tools work by generating and maintaining cryptographic hashes of critical system files and configurations. They periodically compare the current state of files against a known-good baseline to detect unauthorized changes or corruption [8]. In air-gapped systems, FIM is particularly effective in revealing whether malware has modified binaries, startup scripts, or scheduled tasks.

While these tools are very lightweight and ideal for disconnected environments, their primary drawback is that they do not provide real-time prevention or insights into how or why changes occurred. They also generate logs that need to be manually reviewed, which can delay incident response if not automated through local alerting.

Offline Antivirus Solutions

Offline-capable antivirus software performs signature-based scanning without needing an internet connection [9]. These tools are ideal for scanning new USB devices or periodic sweeps of air-gapped systems for known threats.

Their strength lies in their ease of deployment and ability to work independently of cloud services. However, they depend on locally stored virus definitions, which must be updated manually. As a result, they may fail to detect novel or zero-day threats that emerge after the last update.

Sandboxing (Local)

Local sandboxing tools allow suspicious files to be executed in a tightly controlled and isolated environment [10]. This enables analysts to observe behavior like file creation, registry modification, or network simulation without risking the integrity of the host system.

In air-gapped setups, sandboxing offers a safe way to evaluate files introduced via USB or physical media. While powerful, sandboxing typically requires more technical expertise and may be bypassed by malware that employs sandbox-evasion techniques or delays its execution to avoid detection.[11]

USB Activity Monitoring and Control

USB monitoring is particularly valuable in air-gapped systems, where USB devices are often the only means of external interaction [12]. These tools log USB insertions, enforce device access policies, and can even block unauthorized devices altogether.

The primary benefit of USB monitoring is its proactive control over a major attack surface. However, overly restrictive policies can interfere with legitimate user activities, and insider threats using authorized devices may still pose a risk if encryption or hidden partitions are used.

System Behavior Monitoring

Tools like OSSEC or custom-built host auditing scripts can analyze CPU usage, memory spikes, process trees, and file I/O patterns to flag abnormal behaviors [13]. This heuristic analysis helps identify covert channels, tunneling attempts, or resource abuse by stealthy malware.

The flexibility of behavior-based monitoring allows it to detect unknown or obfuscated threats, but tuning such systems is complex and prone to false positives. Moreover, without centralized analytics, response and correlation must be handled manually or via local automation.[14]

Endpoint Detection and Response (EDR) – Local Implementation

(-Nahit: This part was done by Kelly

While many EDR systems rely on cloud services, some can function effectively in offline or air-gapped setups with local logging and analysis. Tools such as Elastic Endpoint, GRR Rapid Response, or custom lightweight EDR agents can record detailed telemetry (e.g., file access, command-line activity, kernel calls) and store them locally for incident analysis.

A key advantage of local EDR is its visibility into system-level activity, allowing correlation across different layers of the operating system. Unlike simple antivirus tools, EDR can uncover multi-stage attacks and lateral movement [15]. However, most local EDRs require significant configuration, storage for logs, and a skilled analyst to interpret the data without cloud-based assistance or threat intelligence feeds [16].

SIEM Agent (Local Implementation)

Security Information and Event Management (SIEM) agents, such as Wazuh or NXLog, can be deployed on air-gapped systems to collect and correlate security logs locally without needing a central server connection [13]. These agents aggregate data from various sources—such as file integrity monitors, endpoint logs, USB activity, and behavior detection systems—enabling a consolidated view of potential threats[17][18].

The primary strength of local SIEM agents is their ability to centralize alerting and incident visibility within an isolated environment, improving operational efficiency and making it easier to identify attack patterns. However, in air-gapped scenarios, these agents operate without real-time updates or access to external threat intelligence feeds. This limits their detection of emerging threats unless the rules and correlation logic are maintained and updated manually.

Behavior Detection (Heuristic Analysis)

Behavior-based detection systems focus on identifying unusual or unauthorized actions rather than relying on static signatures [14]. These tools analyze how applications interact with the system, monitoring patterns like unusual process spawning, elevated privilege usage, or scripting activity that may indicate malware behavior.

In air-gapped environments, behavior detection is valuable for flagging unknown threats that traditional antivirus solutions might miss. Its strength lies in its adaptability against novel attacks and its ability to uncover obfuscated or multi-stage malware. However, these systems can be complex to configure and tune for accuracy, particularly in the absence of centralized alert correlation. False positives are also common, potentially leading to alert fatigue without proper filtering and local analyst involvement.

Application Whitelisting

Application whitelisting restricts system execution to a predefined list of approved software, blocking any unauthorized or unknown programs from running [17]. Tools like Microsoft AppLocker or third-party alternatives are commonly used to enforce these controls at the operating system level.

In air-gapped setups, whitelisting is one of the most effective forms of prevention, as it stops malware before it can even execute. It offers strong protection against both known and unknown threats delivered through USB or other physical media. However, its downside lies in manageability—creating and maintaining an accurate whitelist can be time-consuming, especially in environments with frequent software updates or where user workflows require diverse applications. Improper configuration may block legitimate operations and frustrate users.

Honeypots (Local Use in Air-Gapped Systems)
Honeypots are decoy systems or services deliberately designed to appear vulnerable in order to attract attackers and malware [18]. When deployed locally within an air-gapped network, they can mimic valuable resources—such as file servers, user workstations, or IoT devices—without containing any real data. This allows defenders to observe malicious behavior in a controlled environment and gather intelligence on tools, techniques, and procedures used by adversaries.

In isolated environments, honeypots are particularly useful for detecting insider threats or malware that has somehow infiltrated the system through physical vectors like USB drives. They enable early detection of lateral movement attempts and unauthorized scanning or access patterns. They provide high-fidelity alerts: any interaction with the honeypot is almost certainly malicious. Additionally, they offer unique insights into the behavior of stealthy malware that might otherwise remain dormant on production systems. However, honeypots are passive tools; they do not prevent threats, and sophisticated malware may detect and avoid them. Also, they require careful configuration to avoid accidentally becoming an attack pivot point.

[both]Hardware Versus Software Based Defense

[nahit]While hardware-based defense mechanisms are foundational to enforcing air-gapped security, they often fall short against modern covert channel attacks. Physical solutions like electromagnetic shielding, acoustic dampening, and thermal or infrared monitoring are effective at blocking specific forms of side-channel exfiltration [2][4][5]. For example, data diodes ensure secure one-way communication on the wire, and shielding can prevent emissions from being captured via RF or GSM channels in the air. However, these tools are typically expensive, require complex implementation, and can be

circumvented by newer malware techniques such as NoiseHopper and RAMBO, which exploit advanced frequency hopping and electromagnetic emanations [6][7].

Moreover, these hardware solutions do not scale well for continuous behavioral monitoring or detection. To prevent attacks using hardware solutions, new hardware such as a stronger frequency analyzer, or a thicker wall needs to be utilized to adopt to newer techniques. Adjustments to physical parts of a defense system often prove impractical.

In contrast, software-based detection systems provide adaptive, layered visibility into system behavior, offering a more flexible and cost-effective defense. Offline antivirus software can be easily deployed in air-gapped setups to detect known threats on USB drives or internal storage. Although dependent on manually updated signatures, their simplicity and independence from cloud services make them a staple in offline environments [9].

[kelly]File Integrity Monitoring (FIM) complements antivirus solutions by detecting unauthorized changes to system files or configurations, which can be early indicators of malware presence [8]. FIM tools are lightweight and ideal for systems that rarely change, though they require manual analysis unless paired with local alerting systems.

Among software solutions, Endpoint Detection and Response (EDR) tools that run locally provide the most comprehensive protection. Unlike antivirus programs, EDR solutions like GRR Rapid Response or Elastic Endpoint log detailed system activity (e.g., file access, registry changes, and kernel-level behavior), allowing analysts to investigate sophisticated attacks and lateral movement [15]. Although such systems require more local storage and administrative expertise to operate without cloud-based threat intelligence, their deep visibility makes them highly effective in detecting post-exploitation behavior in air-gapped networks [16].

However, it’s important to note that software-based defense mechanisms should not be blindly trusted. Advanced threat actors are constantly looking for new ways to bypass both existing and emerging defense mechanisms. As a result, protecting critical systems requires a multi-layered approach. This approach is also called defense-in-depth. Meaning, security must be implemented at every possible level. For example, blocking unauthorized access to air-gapped systems, ensuring those systems cannot communicate externally by using data diodes, and using software solutions like File Integrity Monitoring (FIM) to detect malicious activity within the system.

In summary, while hardware-based defenses are essential for blocking external emissions and ensuring physical isolation, they should not be the only relied upon defense. They can be effective in detecting and especially preventing attacks through the use of data diodes and signal monitoring alongside software-based tools such as local EDR and antivirus solutions. Such software-based solutions offer detection of threats after a hardware-based prevention system has been successfully bypassed by an advanced threat or when a hardware-based solution falls short in cases like NoiseHopper. The combined use of both software and hardware-based defense mechanisms provides the strongest resilience in modern air-gapped environments [1][5][15].

Advise

In this section, we outline the key characteristics that must be addressed when creating a secure software-based defense system within an air-gapped environment.

Objective

Design a software-based malware detection system for air-gapped environments, without relying on traditional cloud-dependent antivirus solutions.

Criteria for the Proposed Solution

1. [Nahit]Detection Strategy in Air-Gapped Context

Without Cloud Dependence

- The system must detect malware without relying on traditional antivirus tools that use cloud-based signature databases because air-gapped systems don't have access to data on the internet by definition.
- Emphasis should be placed on **behavior-based detection** and **anomaly detection**, rather than static signature matching because the software cannot regularly update its signature database because of air-gap restrictions.

Without Human Dependence

- The system should stop malicious processes without needing a thumbs up because the approval of a specialist means losing time and data the detected exfiltration process should be stopped before losing any more data.

Catch Malicious Activities in an Air-Gap Context

- The system must be able to detect behavior that is malicious in an air-gap context but is not inherently malicious in a regular computer system (e.g. checking for internet access, writing data to USB drives) as these are common behaviors for airgap malware but should never happen in an air-gapped system.
- The system should check if a process is scanning shared and local directories for documents such as pdf, word, excel because this is a common behavior among air-gap specific malware [26]
- The system should do basic checks against known airgap targeting malware. This should include entire malware samples and parts of software that are commonly used for malicious purposes. [20] The system must prevent autorun as it's commonly used by air-gap targeting malware [37] [38] [39]

- We were advised to improve the security aspect by also checking for suspicious activity during nighttime hours
- The system should do checks/prevents for suspicious activity during nighttime hours, this is important because the airgap system is not connected to the internet and can not send messages to security personnel to stop the exfiltration.
- The system should monitor for Indicators of Compromise (IoCs), such as commands like whoami being executed unexpectedly.

2. [both]Defending Against Covert Channels

A. [kelly]Traditional Covert Channels (USB sticks)

- The system should detect unknown USB or program signatures to detect the presence of unauthorized software in the USB devices used, or to identify any unauthorized USB sticks being connected to the air-gapped system [20].
- The system should detect data being written to a USB from the air-gap, because malware present on an air-gap machine can replicate itself and extract data using USB flash drives to collect data such as credentials, screenshots, and more [20]. This type of attack is often the only way of exfiltration and infection as seen in the section “Previously Detected Malware in Air-Gap Systems”
- The system should be able to detect whether a program or process is trying to detect USB sticks that are connected to the system.it is possible for them to copy itself to the USB stick [20][21]. This check is extremely common amongst air-gap targeting malware as seen in the section “Previously Detected Malware in Air-Gap Systems”

B. [Nahit]Experimental Covert Channels (Electromagnetic, Optical, Magnetic etc.)

- The solution must account for techniques specific to **air-gapped malware exfiltration**, as discussed in the work of **Mordechai Guri**. At minimum, it should include detection logic for:
    - One electromagnetic covert channel (e.g., **Rambo**)
    - One magnetic covert channel (e.g. **Odini**)
- Signal monitoring derivatives such as spectrum analysis, microphones for detecting acoustic signals, cameras for detecting optical signals, are out of scope for software-based detection because it relies on hardware components.
- Monitoring computer components such as the CPU, RAM, power supply, and others, as experimental covert channels may exploit these components to transmit signals such as electromagnetic, magnetic, and other types.

- Derivatives of the Rambo attack all use techniques to bypass the CPU cache. They mostly rely on assembly instructions that provide this behavior. These are called non-temporal memory accesses. One example is the MOVNTI instructure or cache miss attacks that aim to fill the cache with garbage data so that the CPU talks directly to ram.

3. [Nahit]Minimizing Additional Attack Surfaces

- The deployment of the solution should not introduce new vulnerabilities:
    - Pay attention to **supply chain threats**, especially during installation via external media (e.g., USB sticks).
    - Ensure the installation process verifies the integrity of the software (e.g., via digital signatures or checksums).

4. [Nahit]Logging

- The system should also log system behaviors that could relate to these attack types because the software isn't connected to a centralized SIEM. This means that logs need to be inspected independently. Having logs would also allow for easier implementation of a local SIEM solution in the future.

5. [Nahit]Programming Language of Choice

An anti-virus system needs to be fast as it checks for malicious activity in real time and safe as it runs with high privileges. If the program is not safe, then malware could exploit a vulnerability in the anti-virus system to get a higher level of control in the system. For this reason, we moved away from memory-unsafe languages like C and C++. This leaves us with the following common choices: Java, C#, Go, Rust. We excluded Python because it's a slow language that is not well suited for an anti-virus product.

C# is usually Windows specific, and Linux doesn't support every C# feature like Windows does. Java on the other hand can run on both platforms but it depends on the java runtime environment. This causes a dependency in our software that can cause problems in a highly secured air-gapped environment. On top of this, the development experience of developers with Java is often not pleasing. This leaves us with 2 languages: Rust and Go. Rust is an unconventional language that works differently than most other languages. This means that the knowledge gained by learning Rust concepts are not really transferrable to other languages. It's often thought of as a hard language to learn as well. However, Go is a fast, easy-to-learn, and safe language that uses similar concepts to other languages. This gives us a great learning opportunity to explore Go programming which keeps growing in popularity.

[Nahit]Out of Scope

- Detection or prevention of **insider threats**
- Implementation of **AI/ML-based detection models**
- **Hardware-assisted detection mechanisms** (e.g., requiring FPGA or external monitoring devices, external signal monitoring)
- **Internal RAM interference/jamming** methods (due to lack of standardized tools and low practicality)
- Integration with **SIEM software platforms**

[Nahit]Success Criteria

- The system must detect at least **two types** of air-gap exfiltration techniques in a controlled test environment.
- The installation method must include a mechanism to verify the **integrity** of installation media (e.g., via SHA512 checksum).
- The monitoring system must generate alerts within **10 seconds** upon detection of unusual CPU or memory activity.

References

[1] *Air-Gapped systems: the ultimate defense against cybersecurity threats*. (n.d.). https://www.gopher.security/blog/air-gapped-systems-ultimate-defense-against-cybersecurity-threats

[2] *Wikipedia contributors. (2025, February 19). Air gap (networking). Wikipedia. https://en.wikipedia.org/wiki/Air_gap_(networking)*

[3] *What is Air Gap? Essential Guide to Air Gap Security | Fortinet. (n.d.). Fortinet. https://www.fortinet.com/resources/cyberglossary/what-is-air-gap*

[4] *Wikipedia contributors. (2024, August 31). Air-gap malware. Wikipedia. https://en.wikipedia.org/wiki/Air-gap_malware*

[5] *Armis. (2024, November 22). The air gap won't save you | Armis. https://www.armis.com/blog/the-air-gap-wont-save-you/*

[6] Md Faizul Bari, Shreyas Sen (2024, may 6).NoiseHopper: Emission Hopping Air-Gap Covert Side Channel with Lower Probability of Detection. https://engineering.purdue.edu/~shreyas/SparcLab/static/pdfs/c/MB_HOST_NoiseHopper.pdf

[7] M. Guri, "RAMBO: Leaking Secrets from Air-Gap Computers by Spelling Covert Radio Signals from Computer RAM" Nordic Conference on Secure IT Systems

[8] *TripWire Enterprise for secure configuration management and file integrity monitoring*. (n.d.). [Video]. https://www.tripwire.com/

[9] Vigderman, A. (2024, September 17). *How does antivirus software work?* Security.org. https://www.security.org/antivirus/how-does-antivirus-work/

[10] *What is Cuckoo? — Cuckoo Sandbox v2.0.7 Book*. (n.d.). https://cuckoo.readthedocs.io/en/latest/introduction/what/

[11] *What is sandboxing? Sandbox security and environment | Fortinet*. (n.d.). Fortinet. https://www.fortinet.com/resources/cyberglossary/what-is-sandboxing

[12] *USB Device Management Software - USB Device Control & Protection | SYTECA*. (2024, October 22). Syteca. https://www.syteca.com/en/product/usb-blocking

[13] OSSEC. (2025, March 5). *OSSEC - world's most widely used Host Intrusion Detection System - HIDS*. https://www.ossec.net/

[14] SentinelOne. (2024, October 16). *What is Behavior Monitoring? Methods & Strategies*. https://www.sentinelone.com/cybersecurity-101/cybersecurity/what-is-behavior-monitoring/

[15] *What is EDR? Endpoint Detection and Response | Microsoft Security*. (n.d.). https://www.microsoft.com/en-us/security/business/security-101/what-is-edr-endpoint-detection-response?msockid=02930cfd9e7a6d31171f18139f2d6c61

[16] *EDR versus XDR: Wat is het verschil? | Microsoft Beveiliging*. (n.d.). https://www.microsoft.com/nl-nl/security/business/security-101/edr-vs-xdr?msockid=02930cfd9e7a6d31171f18139f2d6c61

[17] **Wazuh.** (z.d.). *Security Information and Event Management (SIEM). Real Time Monitoring*. Geraadpleegd op 19 mei 2025, van https://wazuh.com/platform/siem/

[18] **Stellar Cyber.** (z.d.). *What is SIEM? Components, Capabilities and Architecture*. Geraadpleegd op 19 mei 2025, van https://stellarcyber.ai/learn/what-is-siem/Stellar Cyber

[19] https://owlcyberdefense.com/learn-about-data-diodes/?utm_source=blog&utm_medium=airgap&utm_campaign=airgap-datadiode

[20] Belding, G. (2020, 10 september). *USBCulprit malware: What it is, how it works and how to prevent it*. Infosec Institute. https://www.infosecinstitute.com/resources/malware-analysis/usbculprit-malware-what-it-is-how-it-works-and-how-to-prevent-it/

[21] **Lechtik, M., & Dedola, G.** (2020, 9 juni). *Cycldek: Bridging the (air) gap*. Securelist. https://securelist.com/cycldek-bridging-the-air-gap/97157/

[22] *Sednit espionage group attacking Air-Gapped networks*. (2014, November 11). https://www.welivesecurity.com/2014/11/11/sednit-espionage-group-attacking-air-gapped-networks/

[23] GReAT. (2021, January 13). Sofacy APT hits high profile targets with updated toolset. *Securelist*. https://securelist.com/sofacy-apt-hits-high-profile-targets-with-updated-toolset/72924/

[24] *Ramsay, Software S0458 | MITRE ATT&CK®*. (n.d.). https://attack.mitre.org/software/S0458/

[25] Belding, G. (2025, April 3). *Ramsay malware: What it is, how it works and how to prevent it | Malware spotlight*. Infosec Institute. https://www.infosecinstitute.com/resources/malware-analysis/ramsay-malware-what-it-is-how-it-works-and-how-to-prevent-it-malware-spotlight/

[26] **MITRE.** (2025, 25 april). *USBferry*. MITRE ATT&CK. https://attack.mitre.org/software/S0452/

[27] **Chen, J.** (2020, 12 mei). *Tropic Trooper's Back: USBferry Attack Targets Air-gapped Environments* [Technisch rapport]. Trend Micro. https://documents.trendmicro.com/assets/Tech-Brief-Tropic-Trooper-s-Back-USBferry-Attack-Targets-Air-gapped-Environments.pdf

[28] Shevchenko, S. (2008, 30 november). *Agent.btz – A Threat That Hit Pentagon*. ThreatExpert Blog. https://blog.threatexpert.com/2008/11/agentbtz-threat-that-hit-pentagon.html

[29] Gostev, A. (2014, 12 maart). *Agent.btz: a Source of Inspiration?* Securelist. https://securelist.com/agent-btz-a-source-of-inspiration/58551/

[30] **Kaspersky Global Research & Analysis Team (GReAT).** (2015, 16 februari). *A Fanny Equation: "I am your father, Stuxnet"*. Securelist. https://securelist.com/a-fanny-equation-i-am-your-father-stuxnet/68787/

[31] **Cimpanu, C.** (2017, 22 juni). *Vault 7: CIA has malware for hacking air-gapped networks via USB thumb drives*. BleepingComputer. https://www.bleepingcomputer.com/news/security/vault-7-cia-has-malware-for-hacking-air-gapped-networks-via-usb-thumb-drives/

[32] *China Tick APT group targeting air-gapped systems in Asia*. (n.d.). cyberdefensemagazine.com. https://www.cyberdefensemagazine.com/china-tick-apt-group-targeting-air-gapped-systems-in-asia/

[33] **Gostev, A.** (2012, 28 mei). *The Flame: Questions and Answers*. Securelist. https://securelist.com/the-flame-questions-and-answers/34344/

[34] **Symantec Security Response.** (2012, 28 mei). *W32.Flamer*. Symantec. https://web.archive.org/web/20190930124504/https://www.symantec.com/security-center/writeup/2012-052811-0308-99

[35] **MITRE.** (2025, 16 april). *Flame (S0143)*. MITRE ATT&CK. https://attack.mitre.org/software/S0143/

[36] *PIXHELL Attack: Leaking Sensitive Information from Air-Gap Computers via 'Singing Pixels.'* (n.d.). https://arxiv.org/html/2409.04930v1
[37] *Agent.btz, Software S0092 | MITRE ATT&CK®*. (n.d.). https://attack.mitre.org/software/S0092/

[38] *SHIPSHAPE, Software S0028 | MITRE ATT&CK®*. (n.d.).
https://attack.mitre.org/software/S0028/
[39] *USBStealer, Software S0136 | MITRE ATT&CK®*. (n.d.).
https://attack.mitre.org/software/S0136/