

REPORT DRDO – CYBERSECURITY (CUCKOO OPEN SOURCE)

Title: Exploring the Cuckoo Open Source: Working Parameters and Malware Analysis

Abstract:

This research report delves into the Cuckoo open-source project, a powerful and widely-used tool for malware analysis and detection. By investigating its working parameters and functionality, this study aims to provide a comprehensive understanding of Cuckoo's capabilities and its impact on cyber security. Through an exploration of its architecture, deployment options, and analysis workflow, this report offers valuable insights into how Cuckoo operates and contributes to the field of malware analysis.

1. Introduction

1.1 Background

This section provides an overview of the increasing threat posed by malware and the crucial role of malware analysis in mitigating these risks. It outlines the significance of open-source tools like Cuckoo in addressing the challenges associated with malware detection and analysis.

1.1.1 Overview of the Increasing Threat of Malware

The increasing threat posed by malware has become a significant concern in the digital landscape. Malware, short for malicious software, refers to any software designed with malicious intent to disrupt, damage, or gain unauthorized access to computer systems, networks, or user devices. Here is an overview of the escalating threat posed by malware:

1. **Sophistication:** Malware has evolved over time, becoming more sophisticated and harder to detect. Cybercriminals employ advanced techniques, including encryption, polymorphism, and rootkit functionality, to evade traditional security measures and remain undetected for longer periods.

2. **Diverse Types:** Malware comes in various forms, each with a unique purpose and method of attack. Common types include viruses, worms, Trojans, ransomware, spyware, adware, key loggers, and botnets. Each type targets different vulnerabilities and poses specific risks to individuals, organizations, and critical infrastructure.

3. **Increased Frequency:** The frequency of malware attacks has risen dramatically. With the advent of automation and botnets, cybercriminals can launch large-scale attacks with minimal effort. Malware is distributed through

various means, such as email attachments, malicious websites, infected software, compromised networks, and social engineering tactics.

4. Targeting Individuals and Organizations: Malware attacks target both individuals and organizations. Individuals may fall victim to phishing attempts, drive-by downloads, or infected files, compromising their personal data, financial information, or even identity.

5. Advanced Persistent Threats (APTs): APTs are sophisticated, long-term cyber-attacks orchestrated by well-funded and highly skilled adversaries. They are often associated with nation-state actors, organized crime, or advanced hacking groups.

6. Ransomware Epidemic: Ransomware attacks have witnessed a significant surge in recent years. These attacks involve encrypting the victim's data and demanding a ransom in exchange for the decryption key. High-profile incidents have affected critical infrastructure, healthcare facilities, government agencies, and businesses of all sizes, causing extensive financial losses and disruption of services.

7. IoT Vulnerabilities: The proliferation of the Internet of Things (IoT) devices has introduced new security challenges. Many IoT devices lack robust security measures, making them susceptible to malware exploitation.

8. Supply Chain Attacks: Malware creators are increasingly targeting the software supply chain, compromising trusted vendors and software updates. By injecting malware into legitimate software, cybercriminals can bypass traditional security measures and gain unauthorized access to a wide range of systems or networks.

9. Mobile Malware: With the widespread use of smartphones and mobile devices, malware targeting mobile platforms has grown. Mobile malware can steal sensitive information, track user activities, send unauthorized messages, or perform financial fraud. Malicious apps, app store compromises, and drive-by downloads are common infection vectors.

10. Evolving Threat Landscape: The threat landscape continues to evolve rapidly. New malware variants, attack techniques, and evasion tactics are constantly emerging. Machine learning, artificial intelligence, and other advanced technologies are also being employed by both attackers and defenders, creating a constant cat-and-mouse game in the battle against malware.

To combat the increasing threat posed by malware, organizations and individuals need to adopt a multi-layered security approach, including robust antivirus software, regular software updates, user education, strong authentication practices, network segmentation, and incident response planning.

1.1.2 Significance of Open-Source Tools – Cuckoo

Open-source tools like Cuckoo play a significant role in addressing the challenges associated with malware detection and analysis. Here are some key reasons for their significance:

1. **Flexibility and Customization:** Open-source tools provide a high degree of flexibility and customization options. Security professionals can modify and extend the tool's functionality to meet their specific requirements. This adaptability is crucial in the dynamic landscape of malware, where new variants and attack techniques continually emerge.

2. **Transparency and Trust:** Open-source tools are transparent, allowing security experts to examine the source code, verify its integrity, and ensure that there are no hidden functionalities or vulnerabilities. This transparency builds trust within the security community, as experts can collaborate, contribute, and collectively improve the tool's effectiveness.

3. **Rapid Innovation and Development:** Open-source tools often benefit from a larger community of contributors who can suggest enhancements, fix bugs, and add new features. This collaborative approach enables rapid innovation and development cycles, ensuring that the tool remains up to date with the latest malware trends and analysis techniques.

4. **Cost-Effectiveness:** Open-source tools are typically free to use, making them cost-effective alternatives to proprietary solutions. This accessibility allows organizations with limited budgets or resources to leverage advanced malware detection and analysis capabilities without incurring substantial expenses.

5. **Malware Sample Analysis:** Tools like Cuckoo facilitate the automated analysis of malware samples in controlled environments. They simulate an isolated system where malware can be executed, monitored, and analyzed without posing a risk to the host system. This analysis provides valuable insights into the malware's behavior, capabilities, and potential impact, aiding in its detection and mitigation.

6. **Threat Intelligence Sharing:** Open-source tools encourage the sharing of threat intelligence within the security community. Analysts can exchange information, malware signatures, and behavioral indicators discovered during their analysis, helping others identify and respond to emerging threats more effectively.

7. **Integration and Interoperability:** Open-source tools often offer APIs (Application Programming Interfaces) and support for standard protocols, facilitating integration with other security tools and systems. This interoperability allows for a comprehensive security ecosystem, where different tools can exchange information, share data, and collaborate to provide enhanced malware detection and analysis capabilities.

8. **Education and Skill Development:** Open-source tools provide valuable resources for learning and skill development in the field of malware analysis. Security practitioners, researchers, and students can access the

tool's documentation, tutorials, and community forums to enhance their knowledge and expertise in detecting and analyzing malware.

By leveraging open-source tools like Cuckoo, security professionals can enhance their malware detection and analysis capabilities, stay ahead of evolving threats, and contribute to the overall improvement of cybersecurity practices. These tools democratize access to advanced security technologies and foster collaboration, enabling a collective defense against malware and other cyber threats.

1.2 Research Objectives

This subsection states the specific objectives of the research, including an exploration of Cuckoo's working parameters, its architecture, and the key components involved in its operation. The aim is to provide a comprehensive understanding of how Cuckoo functions as an open-source malware analysis tool.

1.2.1 Exploration of Cuckoo - Working Parameters

Cuckoo is an open-source malware analysis system that provides a platform for automated malware detection and analysis. It utilizes a combination of virtualization, monitoring, and reporting techniques to examine the behavior of malware samples in a controlled environment. Here are the key working parameters of Cuckoo:

1. **Submission:** The analysis process begins by submitting a suspicious file or URL to Cuckoo for analysis. The file can be uploaded directly to the system or provided through other integration methods, such as API calls.
2. **Virtualization:** Cuckoo employs virtualization technologies, such as VirtualBox or VMware, to create isolated environments known as "sandbox" or "guest machines." These virtual machines (VMs) provide a controlled environment where malware samples can be executed without affecting the underlying host system.
3. **Execution and Monitoring:** Once the malware sample is submitted, Cuckoo launches the virtual machine and executes the sample within the isolated environment. During execution, Cuckoo monitors various activities, including file system changes, network traffic, registry modifications, and system processes. It captures a wide range of behavioral data to understand the actions performed by the malware.
4. **Network Capture:** Cuckoo captures network traffic generated by the malware sample to analyze its communication patterns, connections to command-and-control servers, and potential data exfiltration. This information helps in understanding the malware's network behavior and identifying any malicious network activities.
5. **Analysis Modules:** Cuckoo employs different analysis modules to extract additional information from the malware sample. These modules include static analysis, dynamic analysis, and memory analysis. Static analysis examines the file's characteristics without execution, while dynamic analysis focuses on monitoring the runtime

behavior. Memory analysis involves inspecting the memory space used by the malware to identify code injections, anti-analysis techniques, or sensitive information.

6. Reporting: Once the analysis is completed, Cuckoo generates comprehensive reports that summarize the behavior and actions observed during the malware execution. These reports typically include detailed logs, network captures, screenshots, extracted indicators of compromise (IOCs), and any additional findings. The reports help security analysts understand the malware's behavior and potential impact and provide insights for further investigation or mitigation.

7. Integration and Extensibility: Cuckoo is designed to be extensible and supports integration with various security tools and systems. It provides APIs and hooks that allow for customization and integration with external modules, such as antivirus scanners, threat intelligence platforms, or additional analysis tools. This integration enhances the overall analysis capabilities and facilitates sharing of data and intelligence.

Cuckoo's working parameters can be adjusted and customized based on specific analysis requirements. Analysts can configure various settings, such as timeout values, network configurations, resource usage limits, and behavior monitoring options, to optimize the analysis process and adapt to different malware scenarios.

Overall, Cuckoo's working parameters enable the automated analysis of malware samples, providing valuable insights into their behavior, capabilities, and potential risks. By leveraging these parameters, security analysts can efficiently detect and analyze malware, enhance incident response capabilities, and strengthen overall cyber security defenses.

1.2.2 About Cuckoo & Sandbox

A Cuckoo Sandbox is an open-source tool that creates an illusion server just like the host server and launches malware into that mirror server; the idea behind this is that the sandbox fools the malware into thinking it has infected a genuine host.

The sandbox then records the activity of the malware and then generates a report on what the malware has attempted to do and gathers IOC which gives quick and detailed information about how malware behaves.

Cuckoo Sandbox started as a Google Summer of Code project in 2010 within The Honey Net Project.

In computer security, a sandbox is a security mechanism for separating running programs. It is often used to execute untested code, or untrusted programs from unverified third parties suppliers, untrusted users, and untrusted websites.

This concept applies to malware analysis sandboxing too: our goal is to run an unknown and untrusted application or file inside an isolated environment and get information on what it does.

Malware sandboxing is a practical application of the dynamical analysis approach: instead of statically analyzing the binary file to get evidence, it gets executed and monitored in real-time GUI.

1.2.3 Key Components – Cuckoo

Cuckoo, the open-source malware analysis system, consists of several key components that work together to provide automated malware detection and analysis capabilities. Here are the main components of Cuckoo:

1. **Cuckoo Core:** The Cuckoo Core is the central component of the system. It coordinates the overall analysis process, manages the virtual machines (VMs), handles communication with other components, and orchestrates the data flow between them.
2. **Virtualization Technology:** Cuckoo relies on virtualization technologies, such as VirtualBox or VMware, to create isolated environments known as "sandbox" or "guest machines." These virtual machines provide a controlled environment where malware samples can be executed safely without impacting the underlying host system.
3. **Guest Agent:** The Guest Agent is installed within the virtual machine and acts as a bridge between the guest machine and the host system. It facilitates communication with the Cuckoo Core and enables monitoring of the guest machine's activities, such as file system changes, registry modifications, and network traffic.
4. **Analysis Modules:** Cuckoo employs various analysis modules to extract valuable information from the malware samples. These modules include:
 - **Static Analysis:** This module examines the file's characteristics, such as file type, size, and structure, without executing the malware. It can provide insights into file metadata and potential indicators of malicious behavior.
 - **Dynamic Analysis:** The dynamic analysis module focuses on monitoring the runtime behavior of the malware within the isolated environment. It captures and records activities such as system calls, API calls, network traffic, file system changes, and process behavior. This information helps in understanding the malware's behavior and actions.
 - **Memory Analysis:** Memory analysis involves inspecting the memory space used by the malware during execution. It helps identify code injections, anti-analysis techniques, and sensitive information stored in memory. Memory analysis can reveal additional insights into the malware's behavior and potential capabilities.
5. **Network Capture:** Cuckoo captures network traffic generated by the malware sample during its execution. This component allows analysis of the malware's network behavior, including connections to command-and-control servers, data exfiltration attempts, or communication with other malicious entities.
6. **Reporting and Visualization:** Cuckoo generates comprehensive reports summarizing the observed behavior, actions, and findings from the analysis process. These reports typically include detailed logs, network captures,

screenshots, extracted indicators of compromise (IOCs), and other relevant information. Additionally, Cuckoo provides visualization options to help analysts understand the data and make informed decisions.

7. Integrations: Cuckoo is designed to integrate with external tools and systems to enhance its capabilities. It supports integration with antivirus scanners, threat intelligence platforms, additional analysis tools, or custom scripts. These integrations enable the exchange of data, enhance analysis accuracy, and facilitate the sharing of threat intelligence within the security community.

These key components work in conjunction to provide a robust and flexible malware analysis framework. Cuckoo's architecture allows for customization, extensibility, and integration with other security tools, empowering security analysts to efficiently detect and analyze malware samples while improving incident response and strengthening overall cyber security defenses.

1.2.4 Cuckoo Functions – Malware Analysis Tool

Cuckoo is a versatile malware analysis tool that offers a range of functions to assist in the analysis of malicious software. Here are some key functions of Cuckoo as a malware analysis tool:

1. Automated Malware Analysis: Cuckoo provides automated analysis capabilities, allowing users to submit suspicious files or URLs for analysis. Once submitted, Cuckoo executes the malware sample in a controlled environment and captures its behavior, activities, and network interactions.

2. Behavior Monitoring: Cuckoo monitors the behavior of malware samples during execution. It captures system-level events, such as file system changes, registry modifications, process creations, network connections, and API calls. This behavior monitoring helps in understanding the actions performed by the malware and identifying potentially malicious behavior.

3. Network Traffic Analysis: Cuckoo captures and analyzes network traffic generated by the malware sample during execution. It provides insights into the communication patterns, protocols used, destination IP addresses, URLs accessed, and any potential data exfiltration attempts. This information helps in understanding the malware's network behavior and identifying communication with malicious entities.

4. File and Memory Analysis: Cuckoo performs file analysis to extract information about the file's structure, metadata, and embedded resources. It can also analyze memory space used by the malware, which helps in identifying code injections, anti-analysis techniques, and sensitive information stored in memory.

5. Malware Detection: Cuckoo employs various detection techniques, including static and dynamic analysis, to identify malicious characteristics and indicators of compromise (IOCs). It compares the behavior and characteristics of the analyzed sample against known patterns, signatures, and behavioral rules to determine if it is malicious.

6. Reporting and Visualization: Cuckoo generates detailed analysis reports that summarize the observed behavior, activities, and findings of the malware sample. These reports typically include logs, network captures, screenshots, extracted IOCs, and other relevant information. Cuckoo also provides visualization options to help analysts understand and interpret the collected data effectively.

7. Integration with External Tools: Cuckoo supports integration with external security tools and systems. It can integrate with antivirus scanners, threat intelligence platforms, sandbox evasion detectors, or custom scripts. This integration allows for enhanced analysis capabilities, data sharing, and collaboration within the security community.

8. Extensibility and Customization: Cuckoo is designed to be extensible and customizable. It provides APIs, hooks, and configurable options that allow users to customize the analysis process, integrate additional modules, or extend its functionality to meet specific requirements.

Cuckoo's functions make it a powerful tool for automated malware analysis, enabling security professionals to detect, analyze, and understand the behavior of malicious software. Its capabilities assist in threat identification, incident response, and the development of effective mitigation strategies.

2. Literature Review

2.1 Malware Analysis and Detection

This section reviews existing literature on malware analysis and detection techniques, highlighting the importance of automated tools and their role in enhancing cybersecurity practices. It also explores the emergence and significance of open-source solutions in the field.

2.1.1 Existing Malware Analysis - Detection Techniques

There is a substantial body of literature on malware analysis and detection techniques, covering various aspects of the field. Here are some key areas and notable works in the field of malware analysis and detection:

1. Malware Analysis Techniques:

- "Practical Malware Analysis" by Michael Sikorski and Andrew Honig: This book provides a comprehensive introduction to malware analysis techniques, including static and dynamic analysis, code reversing, behavior analysis, and more.
- "Malware Analyst's Cookbook" by Michael Ligh, Steven Adair, Blake Hartstein, and Matthew Richard: This book offers practical guidance and step-by-step instructions for analyzing and dissecting different types of malware samples.

2. Static Analysis Techniques:

- "The Art of Memory Forensics" by Michael Hale Ligh, Andrew Case, Jamie Levy, and Aaron Walters: This book focuses on memory forensics techniques for analyzing malware in memory, including static analysis of memory dumps.

- "Practical Reverse Engineering" by Bruce Dang, Alexandre Gazet, and Elias Bachaalany: This book covers reverse engineering techniques, including static analysis, to understand the inner workings of malware.

3. Dynamic Analysis Techniques:

- "Dynamic Malware Analysis" by Michael Sikorski and Andrew Honig: This book delves into dynamic analysis techniques, such as running malware samples in controlled environments, analyzing network behavior, and monitoring system-level activities.

- "Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software" by Michael Sikorski and Andrew Honig: This book provides practical examples and exercises for dynamic analysis using tools like Cuckoo Sandbox and Wireshark.

4. Machine Learning and AI-based Techniques:

- "Machine Learning and Security: Protecting Systems with Data and Algorithms" by Clarence Chio and David Freeman: This book explores the application of machine learning techniques in malware detection and analysis, including feature extraction, classification algorithms, and anomaly detection.

- "Malware Data Science: Attack Detection and Attribution" by Joshua Saxe and Hillary Sanders: This book focuses on using data science and machine learning to detect, classify, and attribute malware attacks.

5. Threat Intelligence and Detection Systems:

- "Applied Cyber Security and the Smart Grid: Implementing Security Controls into the Modern Power Infrastructure" by Eric D. Knapp and Raj Samani: This book covers techniques for detecting and defending against malware in critical infrastructure systems, including the use of threat intelligence and advanced detection systems.

- "Data-Driven Security: Analysis, Visualization and Dashboards" by Jay Jacobs and Bob Rudis: This book explores data-driven approaches to security, including malware detection, visualization, and creating security dashboards.

2.1.2 Importance of Automated Tools

Automated tools play a crucial role in enhancing cybersecurity practices by providing efficient and effective solutions to address the ever-growing challenges in detecting, analyzing, and mitigating cyber threats. Here are the key reasons highlighting the importance of automated tools in cybersecurity:

1. Speed and Scalability: Automated tools can process vast amounts of data and perform complex tasks at high speed, far surpassing human capabilities. They enable organizations to handle large-scale security operations, such as continuous monitoring, threat detection, and incident response, in real-time or near real-time. This speed and scalability are vital in the face of rapidly evolving cyber threats.

2. Improved Accuracy and Consistency: Automation reduces the risk of human error, which can arise from fatigue, oversight, or lack of attention to detail. Automated tools follow predefined rules and algorithms consistently, ensuring accurate and reliable results. This accuracy and consistency are critical in detecting sophisticated threats and minimizing false positives and false negatives.

3. Enhanced Threat Detection and Response: Automated tools leverage advanced algorithms, machine learning, and behavioral analysis techniques to identify patterns, anomalies, and indicators of compromise. They can quickly detect and respond to known and unknown threats, including malware, phishing attacks, intrusions, and data breaches. Automated threat intelligence gathering and sharing further enhance detection capabilities.

4. Streamlined Incident Response: Automated tools streamline incident response processes by automating routine tasks, such as triaging, data collection, correlation, and reporting. They enable rapid incident identification, investigation, and containment, reducing the mean time to respond (MTTR) and minimizing the impact of security incidents. This accelerates recovery and reduces overall damage.

5. Proactive Vulnerability Management: Automated tools can continuously scan systems, networks, and applications to identify vulnerabilities and misconfigurations. They can assess compliance with security policies and standards and provide actionable recommendations for remediation. By automating vulnerability management, organizations can proactively address security gaps and reduce the window of exposure to potential threats.

6. Resource Optimization: Automation frees up human resources from repetitive and mundane security tasks, allowing security professionals to focus on more complex and strategic activities. It enables them to allocate their expertise and time to threat hunting, analysis, and developing proactive security strategies. This optimization of resources enhances overall cybersecurity effectiveness.

7. Consistent Policy Enforcement: Automated tools ensure consistent enforcement of security policies, configurations, and best practices across an organization's infrastructure and systems. They can enforce access controls, network segmentation, encryption, and other security measures consistently and comprehensively. This reduces the risk of human error, misconfigurations, and policy deviations.

8. Compliance and Auditing: Automated tools simplify compliance management by monitoring and enforcing regulatory requirements, industry standards, and internal policies. They facilitate auditing, logging, and reporting on security controls and events, helping organizations demonstrate compliance during assessments and audits.

In summary, automated tools are essential in enhancing cybersecurity practices by providing speed, accuracy, scalability, and efficiency in threat detection, incident response, vulnerability management, policy enforcement, and compliance. They empower organizations to better protect their digital assets, reduce risk exposure, and adapt to the evolving threat landscape. However, it's important to note that automation should be complemented with human expertise and oversight to ensure effective decision-making and adaptability to emerging threats.

2.1.3 Alternatives of Cuckoo

While Cuckoo is a widely used and highly regarded open-source malware analysis system, there are several alternative tools available in the field of malware analysis and detection. Here are a few notable alternatives to Cuckoo:

- 1. Joe Sandbox:** Joe Sandbox is a comprehensive automated malware analysis platform that offers both on-premises and cloud-based solutions. It provides dynamic analysis, static analysis, and memory analysis capabilities. Joe Sandbox supports a wide range of analysis techniques, including behavior monitoring, code and memory analysis, and network traffic analysis.
- 2. Hybrid Analysis:** Hybrid Analysis is an online malware analysis platform that allows users to upload suspicious files and URLs for analysis. It leverages a combination of both static and dynamic analysis techniques to provide insights into the behavior and characteristics of malware samples. Hybrid Analysis also offers community sharing and collaboration features.
- 3. Any.Run:** Any.Run is a cloud-based malware analysis platform that provides an interactive environment for analyzing malware samples. It allows users to execute files in a controlled environment and observe their behavior in real-time. Any.Run offers extensive monitoring capabilities, including process tracking, network traffic capture, and dynamic behavior analysis.
- 4. VMRay Analyzer:** VMRay Analyzer is a powerful malware analysis solution that combines static and dynamic analysis techniques. It employs hypervisor-based monitoring to capture low-level system activities, enabling deep analysis of malware behavior. VMRay Analyzer supports both on-premises and cloud deployments and offers integration capabilities with other security tools.
- 5. FireEye Malware Analysis:** FireEye offers a range of malware analysis tools and platforms, including FireEye Malware Analysis. It provides advanced malware detection and analysis capabilities, leveraging behavior-based analysis, code emulation, and virtual execution techniques. FireEye's solutions are widely used in enterprise environments for targeted threat detection and incident response.
- 6. SandBlast Analyzer:** SandBlast Analyzer, developed by Check Point Software Technologies, is a comprehensive malware analysis and detection platform. It combines multiple analysis techniques, including sandboxing, static analysis, and threat intelligence, to detect and analyze advanced threats. SandBlast Analyzer offers detailed reports, visualization tools, and integration options.

7. ThreatConnect TC Analyze: ThreatConnect TC Analyze is a malware analysis platform that provides automated analysis capabilities for both malware samples and URLs. It incorporates multiple analysis engines, threat intelligence data, and community sharing features. TC Analyze offers a user-friendly interface for efficient analysis and investigation of potential threats.

2.1.4 Their Advantages & Disadvantages over Cuckoo

Here are the advantages and disadvantages of some alternative tools to Cuckoo in the field of malware analysis and detection:

1. Joe Sandbox:

Advantages:

- Comprehensive analysis capabilities, including dynamic, static, and memory analysis.
- Supports a wide range of analysis techniques and provides detailed reports.
- Offers both on-premises and cloud-based deployment options.

Disadvantages:

- Requires a license or subscription for full access to advanced features.
- May have a steeper learning curve compared to some other tools.

2. Hybrid Analysis:

Advantages:

- User-friendly interface and easy-to-use online platform.
- Offers community sharing and collaboration features.
- Provides insights into malware behavior through a combination of static and dynamic analysis.

Disadvantages:

- Limited to online analysis, which may not be suitable for organizations with strict data privacy requirements.
- Advanced features may require a paid subscription.

3. Any.Run:

Advantages:

- Real-time interactive analysis environment for observing malware behavior.
- Supports dynamic analysis with comprehensive monitoring capabilities.
- Offers a user-friendly interface and a large user community.

Disadvantages:

- Limited reporting and analysis features compared to more comprehensive platforms.
- Online-based analysis may have limitations for sensitive or confidential files.

4. VMRay Analyzer:

Advantages:

- Powerful analysis capabilities with a focus on malware behavior monitoring.
- Supports both on-premises and cloud-based deployments.
- Offers integration options with other security tools.

Disadvantages:

- May require more advanced technical expertise for setup and configuration.
- Higher cost compared to some other tools.

5. FireEye Malware Analysis:

Advantages:

- Advanced malware detection capabilities for targeted threats.
- Offers comprehensive analysis techniques and threat intelligence integration.
- Trusted and widely used in enterprise environments.

Disadvantages:

- Higher cost compared to open-source or community-based tools.
- May require specialized hardware or appliances for deployment.

6. SandBlast Analyzer:

Advantages:

- Comprehensive analysis and detection capabilities leveraging multiple techniques.

- Offers integration options with other security solutions.
- Provides detailed reports and visualization tools.

Disadvantages:

- May require a larger investment and ongoing maintenance.
- Some advanced features may be available only in higher-tier versions.

7. ThreatConnect TC Analyze:

Advantages:

- Offers automated analysis capabilities for malware samples and URLs.
- Integration with the ThreatConnect threat intelligence platform.
- Provides a user-friendly interface for efficient analysis and investigation.

Disadvantages:

- Limited to analysis within the ThreatConnect ecosystem.
- May not have the same level of customization or extensibility as open-source tools.

2.2 Overview of Cuckoo

Here, we present a comprehensive review of the Cuckoo project, including its history, development, and its position within the landscape of malware analysis tools. We examine relevant literature, scholarly articles, and official documentation to establish a foundation for understanding Cuckoo's functionalities.

2.2.1 Cuckoo & Its Working Model

Cuckoo is an open-source malware analysis system designed to provide automated analysis of suspicious files or URLs. It follows a client-server architecture and operates based on a modular and extensible framework. Here is an overview of Cuckoo's working model:

- 1. Submission:** The analysis process begins with the submission of a suspicious file or URL to the Cuckoo system. This can be done through various means, such as a web interface, API, or command-line interface.
- 2. Analysis Environment:** Once a submission is received, Cuckoo sets up a controlled analysis environment, often referred to as a sandbox. The sandbox is a virtualized or isolated environment where the suspicious file or URL is executed.

3. **Instrumentation:** Cuckoo instruments the analysis environment to monitor and capture various activities and behaviors of the malware. This includes monitoring file system changes, registry modifications, network traffic, API calls, and system-level events.

4. **Execution:** The suspicious file or URL is executed within the controlled environment. Cuckoo allows the malware to run and observes its behavior, interactions, and potential malicious actions. During execution, Cuckoo collects and records detailed information about the malware's activities.

5. **Behavioral Analysis:** Cuckoo performs behavioral analysis by monitoring the dynamic behavior of the malware. It captures and logs the actions performed by the malware, such as file creation, network connections, process creation, and registry modifications. This behavioral analysis helps in understanding the intentions and capabilities of the malware.

6. **Network Traffic Analysis:** Cuckoo captures and analyzes the network traffic generated by the malware during its execution. It monitors the network connections made by the malware, records the communication protocols used, destination IP addresses, URLs accessed, and data transferred. This analysis provides insights into the network behavior of the malware and helps identify potential command-and-control (C2) servers or data exfiltration attempts.

7. **Code and Memory Analysis:** Cuckoo also performs code and memory analysis to gain a deeper understanding of the malware's inner workings. It may analyze the file's structure, unpacking mechanisms, encryption techniques, code injections, and any anti-analysis measures employed by the malware. Memory analysis helps in identifying runtime modifications, code injections, and volatile artifacts left by the malware.

8. **Reporting and Analysis:** Once the analysis is completed, Cuckoo generates a detailed analysis report. The report includes captured data, logs, screenshots, network captures, behavioral indicators, extracted IOCs (Indicators of Compromise), and any additional information collected during the analysis. This report helps analysts understand the behavior and impact of the malware and assists in incident response and mitigation.

9. **Integration and Extensibility:** Cuckoo provides an extensible framework that allows integration with external tools, systems, or custom modules. This enables users to enhance the analysis capabilities, incorporate additional analysis techniques, integrate threat intelligence, or automate workflows.

Cuckoo's working model provides an automated and scalable approach to malware analysis. By combining behavioral analysis, network traffic analysis, code and memory analysis, and comprehensive reporting, Cuckoo assists in identifying and understanding the behavior and impact of malicious software, enabling organizations to respond effectively to cyber threats.

2.2.2 IOC Model – Cuckoo

Indicators of compromise (IOCs) refer to data that indicates a system may have been infected by a cyber threat. They provide cyber security teams or SOC(Security Operations Centers) with crucial knowledge after a data breach or another breach in security.

An IOC is cyber-equivalent evidence left at a crime scene. They're essentially digital versions of tire tracks, fingerprints, and broken windows

Types of IOCs –

- Unusual Network Traffic
- Privileged User Logins from Foreign Countries
- Strange DNS Requests
- System File Changes
- Use of a Suspicious Application Programming Interface (API)

2.2.3 IOC vs IOA Model

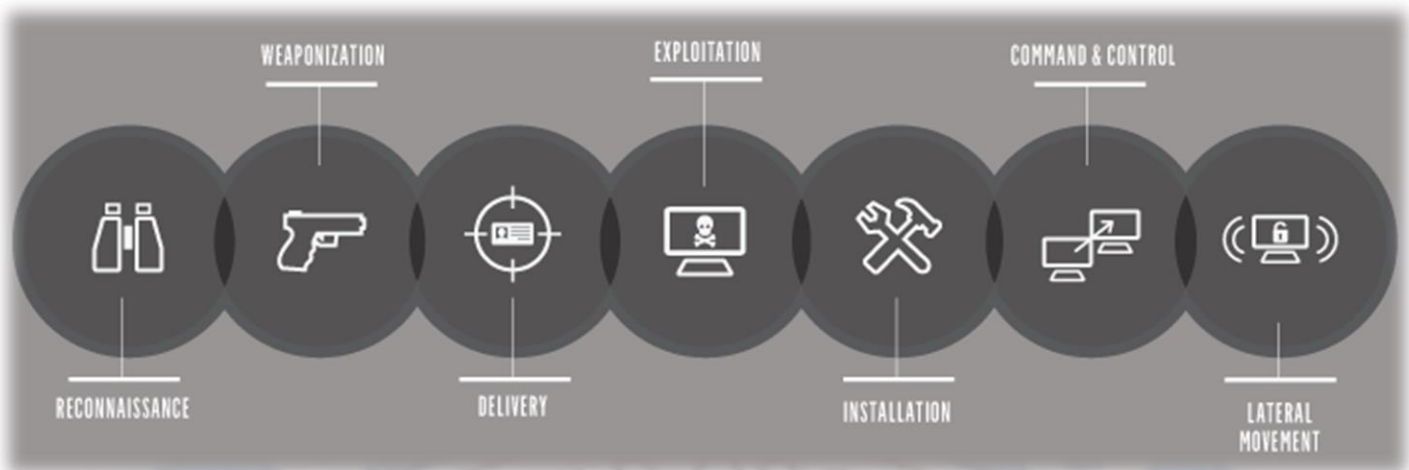
Indicators of attack (IOA) focus on detecting the intention of an attacker that he is trying to accomplish, regardless of the malware or exploit used in an attack.

Indicator of Compromise (IOC) gathers the evidence that some security breach has been done so that smarter tools can be made to detect that malicious file in the future.

Cuckoo Sandbox is based on IOC but next-generation tools need IOA to reduce security breaches.

Example:

- So suppose a robber tried for bank robbery and succeeded so IOA will gather information like he parked the car then enters the bank then tried to open the vault. That makes a report on all the actions he performed so that in the future the security team becomes alert if somewhat happens like this.
- Whereas IOC will give info about what a CCTV will give like the clothes of the robber, what he has used to open the vault, etc. But by this, we can't stop it in future maybe someone with a different car or clothes can rob the bank again.
- Making it to cyber most prominent threat is phishing so actions like clicking on a random mail and downloading a file and then that malware will approach to control and command panel to execute and steal confidential information.



2.2.4 Sandbox Model & Components

In computer security, a sandbox is a security mechanism for separating running programs. It is often used to execute untested code, or untrusted programs from unverified third parties, suppliers, untrusted users, and untrusted websites.

This concept applies to malware analysis sandboxing too: our goal is to run an unknown and untrusted application or file inside an isolated environment and get information on what it does. Malware sandboxing is a practical application of the dynamical analysis approach: instead of statically analyzing the binary file to get evidence gets executed and monitored in real-time GUI.

Here are the key components of the sandbox model:

1. **Virtualization/Isolation Environment**: The sandbox relies on virtualization or isolation technologies to create a separate environment where the malware can run. Virtualization technologies like VMware, VirtualBox, or containerization platforms like Docker are commonly used to create an isolated virtual machine or container.

2. **Operating System**: The sandbox environment includes an operating system that mimics the target system where the malware is intended to execute. It could be a full-fledged operating system or a lightweight version optimized for analysis purposes.

3. **System Monitoring and Instrumentation**: The sandbox monitors and instruments various aspects of the sandboxed environment to capture and analyze the behavior of the malware. It may include components like system-level hooks, API monitors, file system monitors, network sniffers, and registry monitors.

4. **Dynamic Analysis**: During the execution of malware within the sandbox, dynamic analysis techniques are employed to observe and capture the runtime behavior of the malware. This includes monitoring system calls, file system modifications, registry changes, network connections, process creation, and other interactions with the environment.

5. Network Analysis: The sandbox may include network analysis components to capture and analyze the network traffic generated by the malware. This helps in identifying communication with command-and-control servers, data exfiltration attempts, or other malicious network activities.

6. Code and Memory Analysis: The sandbox may perform code and memory analysis techniques to gain deeper insights into the inner workings of the malware. This could involve techniques like code emulation, code injection, unpacking, and examining runtime modifications. Memory analysis helps in identifying injected code, persistence mechanisms, and other volatile artifacts.

7. Behavioral Analysis and Heuristics: The sandbox employs behavioral analysis techniques to identify suspicious or malicious behavior exhibited by the malware. It may use predefined rules, signatures, or heuristics to detect indicators of compromise (IOCs) and potential malicious activities.

8. Reporting and Analysis: The sandbox generates a detailed analysis report summarizing the behavior, activities, and potential impact of the malware. The report may include logs, captured network traffic, screenshots, extracted IOCs, and any additional findings from the analysis process. This report helps analysts understand the nature of the malware and take appropriate mitigation actions.

The sandbox model and its components allow for the safe and controlled execution of malware, enabling security analysts to analyze and understand the behavior and impact of malicious software. By providing an isolated environment, monitoring capabilities, and analysis techniques, sandboxes play a crucial role in malware analysis and detection.

3. Cuckoo Architecture

3.1 Overview of Cuckoo's Architecture

This section provides an in-depth analysis of the architectural framework of Cuckoo. We explore its modular structure, component interactions, and how it integrates with other tools and technologies. A detailed explanation of the architecture sets the stage for understanding Cuckoo's working parameters.

3.1.1 Architectural Framework of Cuckoo

The architectural framework of Cuckoo, an open-source malware analysis system, follows a client-server model. It consists of various components that work together to facilitate the analysis and reporting of suspicious files or URLs. Here are the key components of Cuckoo's architectural framework:

1. **Cuckoo Server:** The Cuckoo Server is the core component of the architecture. It manages and coordinates the entire analysis process. It receives analysis requests from clients, schedules and assigns tasks, and orchestrates the interaction between different components. The server also stores analysis results and generates reports.

2. **Cuckoo Web Interface:** The Cuckoo Web Interface provides a user-friendly interface for interacting with the Cuckoo Server. It allows users to submit files or URLs for analysis, monitor analysis progress, view analysis reports, and manage analysis configurations. The web interface simplifies the user experience and facilitates the management of analysis tasks.

3. **Cuckoo Agent:** The Cuckoo Agent is installed on the analysis machines or virtual machines where the actual malware analysis takes place. The agent is responsible for executing the submitted files or URLs within the controlled environment and collecting relevant information about their behavior and activities. It communicates with the Cuckoo Server to provide real-time updates on the analysis progress.

4. **Virtualization/Isolation Environment:** Cuckoo leverages virtualization or isolation technologies to create a secure and isolated environment for malware analysis. It may use hypervisors, such as VirtualBox or VMware, or containerization platforms like Docker to set up the analysis machines or virtual machines where the malware is executed. This ensures that the malware does not affect the host system or network.

5. **Analysis Modules:** Cuckoo supports various analysis modules that perform specific tasks during the analysis process. These modules include behavior analysis, network analysis, code and memory analysis, and other analysis techniques. Each module is responsible for monitoring and capturing specific aspects of the malware's behavior and generating relevant analysis data.

6. **Database:** Cuckoo uses a database to store analysis results, configuration settings, and other relevant information. The database allows for efficient storage, retrieval, and management of analysis data. It also enables historical analysis data to be accessed and used for comparison or further analysis.

7. **Reporting Module:** Cuckoo includes a reporting module that generates detailed analysis reports. The module processes the captured data from different analysis modules and compiles it into a comprehensive report. The report includes information about the malware's behavior, network communication, code analysis findings, and any other relevant data collected during the analysis process.

8. **Integration and Extensibility:** Cuckoo provides an extensible framework that allows for integration with external tools, systems, or custom modules. This enables users to enhance the analysis capabilities, integrate with existing security infrastructure, or customize the analysis workflow to meet specific requirements.

Overall, the architectural framework of Cuckoo combines client-server communication, virtualization or isolation environments, analysis modules, a database, and a reporting module to facilitate the automated analysis of suspicious files or URLs and provide detailed reports on their behavior and impact.

3.1.2 Other Tools and Technologies

Cuckoo is an open-source automated malware analysis system that helps in analyzing suspicious files and detecting potential threats. In addition to its core functionality, there are several other tools and technologies that complement and enhance the capabilities of Cuckoo. Here are some of them:

1. **Yara:** Yara is a powerful pattern matching tool used to identify and classify malware. Cuckoo integrates with Yara to provide signature-based detection capabilities. Yara rules can be created to define specific patterns or characteristics of malware, and Cuckoo can utilize these rules to flag potentially malicious files during analysis.
2. **Suricata:** Suricata is a network intrusion detection and prevention system (IDS/IPS). It can be integrated with Cuckoo to monitor network traffic during malware analysis. Suricata helps detect and analyze network-based attacks and provides additional insights into the behavior of the malware.
3. **Volatility:** Volatility is a popular open-source memory forensics framework used for analyzing memory dumps. Cuckoo leverages Volatility to extract valuable information from memory snapshots captured during malware analysis. This information can include details about running processes, open network connections, injected code, and other artifacts that aid in malware analysis.
4. **Wireshark:** Wireshark is a widely used network protocol analyzer. Cuckoo can integrate with Wireshark to capture and analyze network traffic generated by malware samples. This allows analysts to understand how the malware communicates over the network, including any malicious activities such as command-and-control communication or data exfiltration.
5. **IDA Pro:** IDA Pro is a professional disassembler and debugger commonly used for reverse engineering purposes. Cuckoo can integrate with IDA Pro to provide detailed static analysis of the malware's code. By analyzing the disassembled code, analysts can gain insights into the malware's behavior, identify malicious functions, and understand its overall structure.
6. **Viper:** Viper is a binary analysis and management framework. It can be used to store, organize, and analyze malware samples. Cuckoo can integrate with Viper to automatically submit and retrieve malware samples for analysis. Viper provides a centralized repository for storing and managing malware samples, making it easier to track and analyze a large number of samples.

These are just a few examples of the tools and technologies that can be used in conjunction with Cuckoo to enhance malware analysis capabilities. The integration of these tools allows for a more comprehensive and in-depth analysis of potentially malicious files, aiding in the identification and mitigation of cyber threats.

3.1.3 Files Cuckoo Can Work Upon

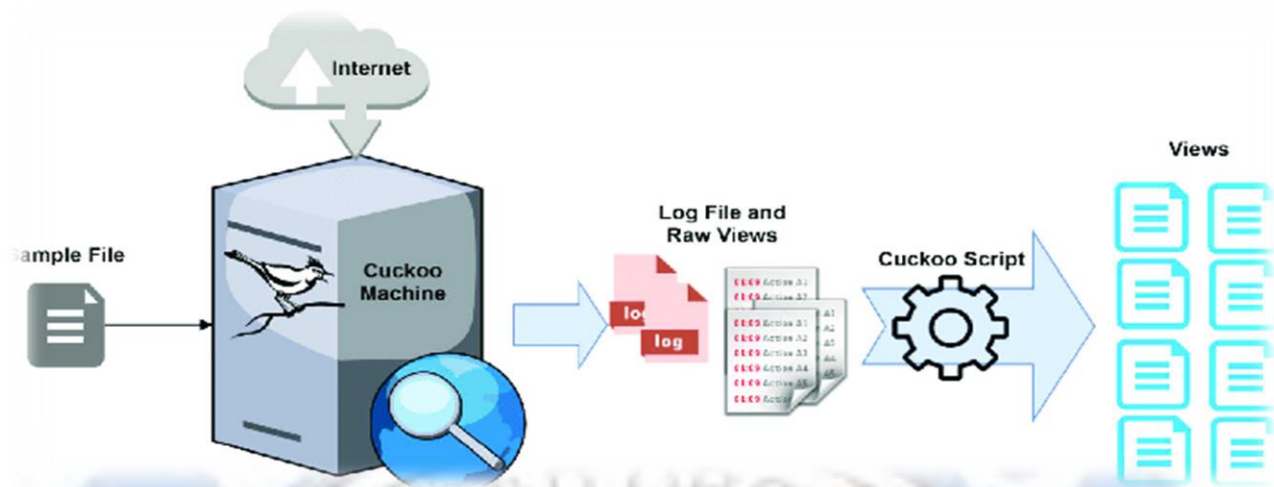
It can be used to analyze:

- ☐ Generic Windows executable
- ☐ DLL files
- ☐ PDF documents
- ☐ Microsoft Office documents
- ☐ URLs and HTML files
- ☐ PHP scripts
- ☐ CPL files
- ☐ Visual Basic (VB) scripts

- ☐ ZIP files
- ☐ Java JAR
- ☐ Python files

3.1.4 Cuckoo Components

- The main components of Cuckoo's infrastructure are a Host machine (the management software) and a number of Guest machines (virtual or physical machines for analysis).
- The Host runs the core component of the sandbox that manages the whole analysis process, while the Guests are the isolated environments where the malware samples get actually safely executed and analyzed.
- Cuckoo makes a detailed report and sent back to the host server.



3.1.5 What Can Cuckoo Do?

- Trace API calls and general behavior of the file and distill this into high-level information and signatures comprehensible by anyone.
- Dump and analyze network traffic, even when encrypted with SSL/TLS. With native network routing support through InetSIM, a network interface, or a VPN in PCAP format
- Perform advanced memory analysis of the infected virtualized system through Volatility as well as on a process memory granularity using YARA.
- Files are created, deleted, and downloaded by the malware during its execution.
- Screenshots are taken during the execution of the malware.
- Full memory dumps of the machines.

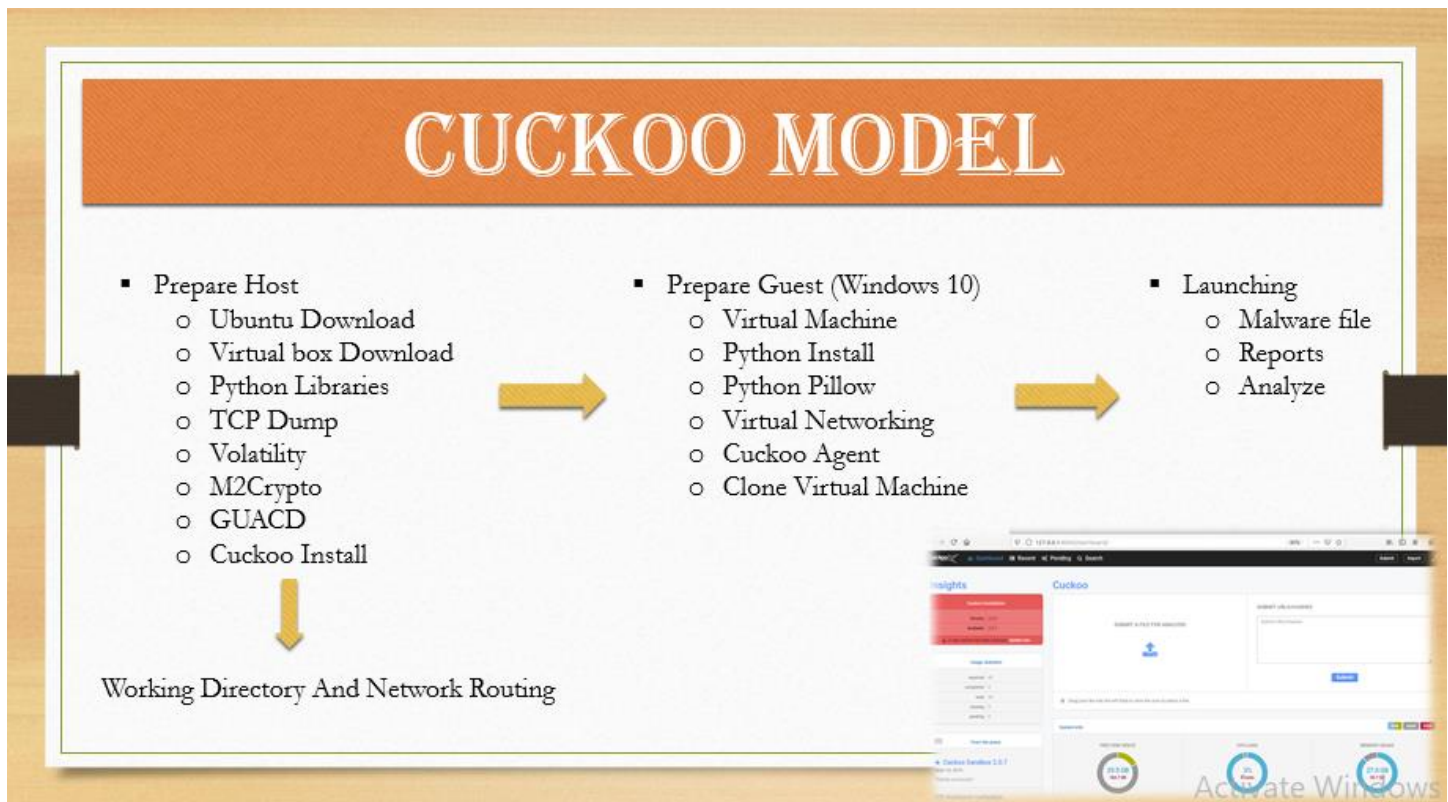
4. Working Parameters of Cuckoo

This section provides an analysis of the working model of the Cuckoo. We explore its modular structure, component interactions, and how it integrates with other tools and technologies. A detailed explanation of Cuckoo's working parameters.

4.1 Installation and Configuration

Here, we discuss the installation process and configuration options of Cuckoo. We provide step-by-step guidance on setting up the necessary dependencies, configuring the system environment, and fine-tuning the parameters to optimize its performance.

4.1.1 Cuckoo Setup Model



Setting up Cuckoo for malware analysis involves several steps, including the installation of necessary dependencies, configuring the Cuckoo environment, and preparing the host and guest systems for analysis. Here's a general overview of the steps involved in setting up Cuckoo:

1. Prepare the Host System:

- Choose a dedicated host system (physical or virtual) with a supported operating system (e.g., Ubuntu, Debian, CentOS).
- Install the required dependencies, such as Python, MongoDB, and other system libraries.
- Configure the network interfaces and firewall settings to enable communication with the guest systems.

2. Prepare the Guest System:

- Choose a guest system to run the malware samples (e.g., Windows 7, Windows 10).
- Install the guest system and configure it with the necessary software and updates.
- Take a snapshot or create a clean backup of the guest system to revert to a pristine state after each analysis.

3. Install Cuckoo Dependencies:

- Install Cuckoo's dependencies, including virtualization software (e.g., VirtualBox, VMware), Volatility, and other required Python packages.
- Configure the virtualization software to allow Cuckoo to control the guest systems.

4. Configure Cuckoo:

- Set up the Cuckoo environment by creating a configuration file that specifies various settings, such as network configuration, analysis options, and submission interfaces.
- Configure the analysis options, such as the timeout values, behavior analysis modules, and logging settings.

5. Configure Cuckoo Reporting:

- Set up the reporting options to specify how analysis results should be stored and presented (e.g., MongoDB, SQLite, JSON, Elasticsearch).

6. Configure Additional Tools:

- Configure any additional tools and technologies you want to integrate with Cuckoo, such as Yara, Suricata, IDA Pro, or Wireshark.

7. Test and Verify:

- Test the Cuckoo setup by analyzing a benign file and verifying that the analysis is performed correctly.
- Ensure that the guest system resets to a clean state after each analysis.

8. Fine-tune and Secure the Setup:

- Adjust Cuckoo's configuration and behavior based on your specific requirements and environment.
- Implement security measures, such as sandbox isolation, network segregation, and access controls, to protect the host and other systems from potential malware threats.

It's important to note that the exact steps and configurations may vary depending on the specific version of Cuckoo, operating systems, virtualization software, and additional tools you choose to integrate. Therefore, it's recommended to refer to the official documentation and resources provided by the Cuckoo project for detailed instructions tailored to your specific setup.

Written Resources-

- <https://cuckoo.readthedocs.io/en/latest/installation/host/installation/>
- <https://hatching.io/blog/cuckoo-sandbox-setup/>

- <https://medium.com/@oshara.16/setting-up-cuckoo-sandbox-for-dummies-malware-analysis-3daa99e950b5>
- <https://www.varonis.com/blog/cuckoo-sandbox>

Video Resources-

- <https://youtu.be/FsF56772ZvU>
- https://youtu.be/QlQS4gk_lFU
- <https://youtu.be/2aSvEC-wdRk>

4.1.2 Cuckoo Working Directory & Components

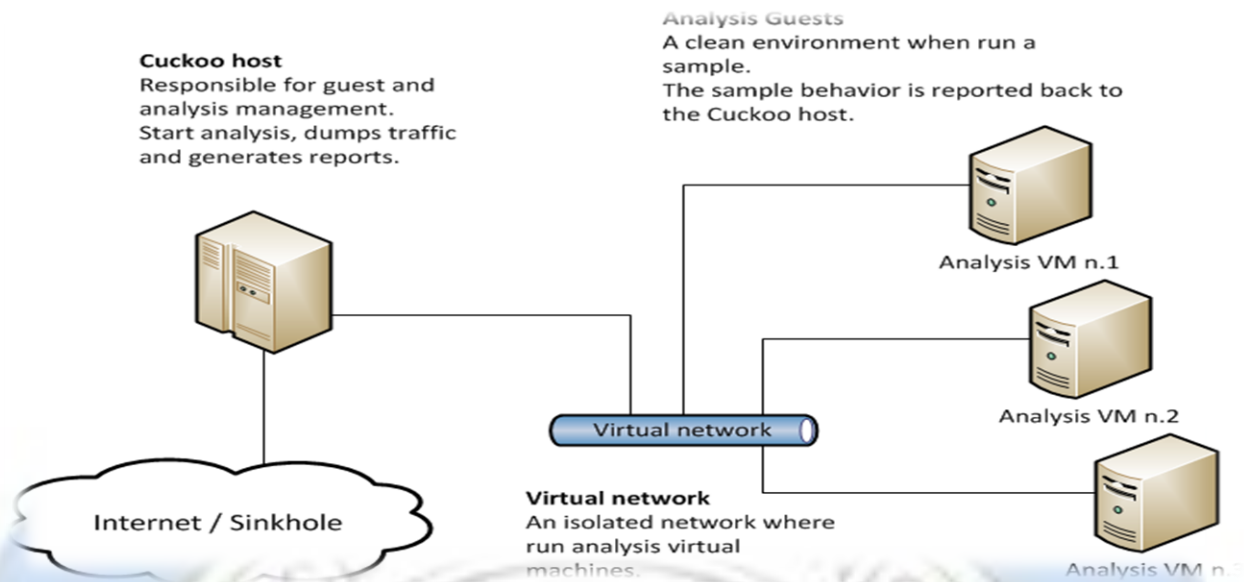
- ☐ Configuration - conf. This file allows you to enable, disable and configure all processing modules. These modules are located under the cuckoo.
 - cuckoo.conf: for general behavior and analysis
 - <machinery>.conf: showing options of virtualization software
 - memory.conf: Volatility configuration.
 - reporting.conf: for enabling or disabling report formats.
- ☐ Cuckoo Signatures
- ☐ Cuckoo Analyzer
- ☐ Cuckoo Agent
- ☐ Yara rules
- ☐ Cuckoo Storage (shows analysis results)

4.1.3 Machinery Modules

This option defines which Machinery module you want Cuckoo to use to interact with your analysis machines. The value must be the name of the module without extension (e.g. Virtual box or VMware).

Oracle VM Virtual Box is cross-platform virtualization software. It allows users to extend their existing computer to run multiple operating systems including Microsoft Windows, Mac OS X, Linux, and Oracle Solaris, at the same time.

Whereas my Host Server is Ubuntu 22.0 version (Ubuntu is a Linux distro based on Debian. It is suitable for cloud computing, servers, desktops, and internet of things (IoT) devices).



4.1.4 Volatility Configuration

- ☐ The Volatility Framework is an open source of tools, implemented in Python, analysts use this for extracting artifacts from memory dumps of (RAM).
- ☐ Volatility Workbench is a graphical user interface (GUI) for the Volatility tool. Volatility is a command line memory analysis of memory dumps that indicates a data breach.
- ☐ A memory dump is a process of taking all information from RAM and copying it to a storage drive as a memory dump file.
- ☐ Developers commonly use memory dumps (also called core dumps) to gather diagnostic information at the time of a crash that tells about the event. Information gathered from the memory dump can help developers fix errors in operating systems and other programs of all kinds.

4.1.5 Parameters Used by Cuckoo

Some of the parameters and techniques used by cuckoo sandbox for malware detection and analysis include:

- ☐ Behavioral Analysis
- ☐ Network Traffic Analysis
- ☐ Static Analysis
- ☐ Dynamic Analysis
- ☐ Signature-based Detection
- ☐ Heuristics and Machine Learning

By combining these techniques, cuckoo sandbox provides an automated and comprehensive analysis of files, helping in the detection and identification of malware.

4.2 Detail Analysis of Parameters

This section provides an in-depth analysis of the Parameters of the Cuckoo. We explore its modular structure, component interactions, and how it integrates with other tools and technologies. A detailed explanation of Cuckoo's working parameters.

4.2.1 Behavioral & Network Traffic Analysis

Cuckoo sandbox monitors the behavior of the file, such as file system activity, network traffic, process monitoring, and registry changes. It checks for suspicious activities that could indicate the presence of malware.

Cuckoo sandbox captures and analyzes network traffic generated by the file being analyzed. It can detect communication with known malicious IP addresses or domains, abnormal network behavior, or connections to known command-and-control servers.

4.2.2 Static & Dynamic Analysis

Cuckoo sandbox performs static analysis on the file without executing it. It examines the file's structure, metadata, and characteristics to identify potential indicators of malware, such as malicious signatures, packer or obfuscation techniques, or suspicious code patterns.

Cuckoo sandbox executes the file in a controlled environment and monitors its runtime behavior. It analyzes system calls, API interactions, and other runtime events to identify any malicious or suspicious actions performed by the file.

4.2.3 Signature-based Detection & Heuristics and Machine Learning

Cuckoo Sandbox can compare the file against a database of known malware signatures. If the file matches any known signatures, it indicates the presence of malware.

The cuckoo Sandbox may employ heuristics and machine learning techniques to identify potentially malicious behavior based on patterns, anomalies, or statistical models. This approach helps detect new or unknown malware that may not have specific signatures.

4.2.4 GUARD and M2Crypto

- GUARD is a web application security library developed by the (OWASP). It stands for "Generalized Architecture for Dynamic Infrastructure Defense" and aims to provide a comprehensive set of security controls and mechanisms to protect web applications from various types of attacks, including injection attacks, cross-site scripting (XSS), cross-site request forgery (CSRF), and more.
- M2Crypto is a Python library that provides cryptographic and SSL/TLS functionalities. It allows developers to incorporate encryption, digital signatures, SSL/TLS protocols, and certificate management into their Python applications. M2Crypto is built on top of OpenSSL, which is a widely used open-source cryptographic library. With M2Crypto, developers can secure communication channels, and perform secure data encryption and decryption.

4.2.5 Cuckoo Signatures & Analyzer

- ❑ Cuckoo Signatures are a set of rules or patterns used by Cuckoo Sandbox to detect specific behaviors of malware or suspicious files. These signatures are similar to antivirus signatures or indicators of compromise (IOCs). They can be based on various attributes, such as file hashes, network connections, API calls, registry modifications, or other behavioral patterns.
- ❑ The Cuckoo Analyzer is a core component of Cuckoo Sandbox responsible for analyzing and extracting information from the submitted files. It employs various techniques, such as behavioral analysis, network traffic monitoring, and dynamic and static analysis, to gain insights into the file's behavior and potentially malicious activities. It facilitates the identification and classification of malware.

4.2.6 YARA Framework

- ✓ The YARA framework is an open-source tool used for malware identification and classification. It stands for "Yet Another Recursive Acronym." It allows security researchers, malware analysts, and incident response teams to create custom rules to identify and categorize files based on specific patterns, strings, or behavioral characteristics.
- ✓ These rules can include strings, regular expressions, Boolean logic operators, and metadata.
- ✓ The YARA framework consists of two main components:
 - YARA Rule Compiler
 - YARA Engine

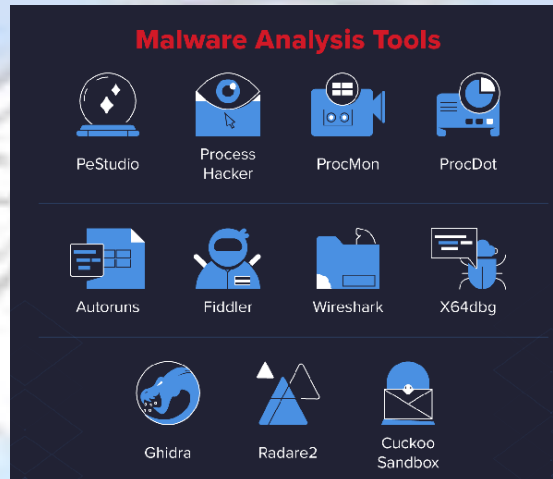
4.2.7 Python Pillow & PE Studio

- ❑ The pillow is a popular open-source Python library used for image processing and manipulation. It serves as a powerful tool for working with images in various formats, including opening, saving, modifying, and enhancing images. Pillow provides a wide range of functionalities such as resizing, cropping, rotating, filtering, adding text or shapes, and applying various image effects.
- ❑ PE Studio is a commercial software tool used for analyzing and examining Windows executable files, specifically Portable Executable (PE) files. PE files are the binary file format. It helps in the analysis of PE files to identify potential security risks and malware. It provides detailed information about the file's structure, dependencies, imported and exported functions, and other attributes.

4.2.8 Network Routing - TCP Dump/TOR/VPN

- ❑ Network routing is the process of determining the path that network traffic should follow from its source to its destination across an interconnected network. It involves making decisions at routers and switches to forward packets based on routing tables and protocols.
- ❑ TCP dump is a command-line packet sniffing tool used to capture and analyze network traffic in real time. TCP dump captures packets at the operating system level, providing detailed information about the source and destination IP addresses, protocols, ports, packet payloads, and other relevant data. It is commonly used for network troubleshooting, network security analysis, and protocol debugging.

- ❑ **TOR (The Onion Router):** TOR is a free and open-source software project that enables anonymous communication over the Internet. It creates a layered and encrypted network path, thereby anonymizing the origin and destination of internet traffic. TOR helps protect user privacy by concealing their IP addresses.
- ❑ **VPN (Virtual Private Network):** It is a technology that creates a secure and encrypted connection between a user's device and a remote network or server. It allows users to establish a private and encrypted tunnel over a public network, such as the Internet. VPNs are commonly used to enhance privacy and secure data transmission.



4.3 Analysis Workflow

This subsection describes the workflow of Cuckoo, from the submission of suspicious files to the generation of analysis reports. We examine the various stages involved, such as sample execution, behavioral analysis, network monitoring, and post-analysis activities. The working parameters of Cuckoo at each stage are elucidated.

4.3.1 Workflow of Cuckoo

The workflow of Cuckoo typically involves several steps to analyze a suspicious file or URL and generate a report with the findings. Here's a high-level overview of the typical workflow:

1. Submission:

- The analyst submits a suspicious file or URL to Cuckoo for analysis.
- Cuckoo receives the submission and begins the analysis process.

2. Virtual Machine Setup:

- Cuckoo selects an available guest virtual machine (VM) or spins up a new VM if needed.
- The VM is prepared by resetting it to a clean state using a snapshot or backup.

3. Instrumentation:

- Cuckoo configures the guest VM to enable monitoring and control.
- Instrumentation involves installing necessary agents and hooks in the VM to collect behavioral data and capture system activities.

4. Execution:

- The suspicious file is executed in the instrumented guest VM.
- Cuckoo monitors the behavior of the file, including file system activity, network communication, and system calls.

5. Behavioral Analysis:

- Cuckoo captures and records the behavior of the file during execution.
- Various analysis modules are triggered to extract relevant information, such as file modifications, registry changes, network traffic, and spawned processes.
- Behavioral analysis modules may include signature-based detection, anomaly detection, and heuristics.

6. Network Monitoring:

- If network monitoring tools (e.g., Wireshark, Suricata) are integrated, Cuckoo captures network traffic generated by the suspicious file.
- Network traffic analysis helps identify communication with malicious domains or command-and-control servers.

7. Memory Analysis:

- Cuckoo may perform memory analysis using tools like Volatility.
- Memory analysis allows for the examination of processes, loaded modules, and potential code injections by the suspicious file.

8. Reporting:

- Cuckoo generates a comprehensive report summarizing the analysis results.
- The report includes details about observed behavior, network connections, file modifications, created processes, and any detected malicious indicators.
- If integrated, additional tools like Yara may be used to provide signature-based detections in the report.

9. Cleanup:

- Once the analysis is complete, Cuckoo cleans up the guest VM, restoring it to the initial clean state.
- The VM is ready for the next analysis.

10. Review and Action:

- The analyst reviews the generated report and identifies potential threats or malicious behavior.
- Based on the findings, appropriate actions can be taken, such as blocking malicious URLs or IP addresses, updating security measures, or further investigating the malware.

4.3.2 Generation of Analysis Reports

In Cuckoo, the generation of analysis reports is an essential part of the workflow, as it provides a comprehensive summary of the analysis results. Cuckoo offers various reporting options that can be configured to suit specific requirements. Here's an overview of the process involved in generating analysis reports:

1. Analysis Completion:

- Once the analysis of a suspicious file or URL is complete, Cuckoo gathers all the collected data and analysis results.

2. Report Configuration:

- Cuckoo allows the configuration of reporting options through the Cuckoo configuration file.
- The configuration specifies the format and destination for the analysis reports.

3. Report Generation:

- Cuckoo utilizes the configured reporting options to generate the analysis report.
- The report typically includes a detailed overview of the analysis results, including behavioral information, network activity, and identified indicators of compromise (IOCs).

4. Report Content:

- The generated analysis report may include the following information:
 - General information about the analysis, such as the submission date and time, file or URL details, and the analysis duration.
 - Summary of behavioral analysis findings, including file modifications, registry changes, processes created, and network connections made.
 - Network traffic analysis, including details of HTTP requests, DNS queries, and any identified malicious communication.
 - Memory analysis findings, such as suspicious processes, injected code, or loaded modules.

- Signature-based detections, if integrated with tools like Yara or other antivirus engines.
- Any additional information or artifacts relevant to the analysis, such as screenshots, extracted files, or dropped malware samples.

5. Report Formats:

- Cuckoo supports various reporting formats, including but not limited to:
 - HTML: A comprehensive report in HTML format that can be easily viewed in a web browser.
 - JSON: A structured report in JSON format that allows for easy integration with other tools or systems.
 - XML: An XML-based report format suitable for further processing or integration with other systems.
 - SQLite: A report stored in a SQLite database, allowing for advanced querying and analysis.

6. Report Delivery:

- The analysis report can be delivered through different channels based on the configuration.
- Common delivery options include storing the report in a specified directory, sending it via email, or integrating it with a ticketing or notification system.

7. Report Review and Action:

- The generated analysis report is typically reviewed by analysts or security professionals.
- Based on the findings in the report, appropriate actions can be taken, such as implementing security measures, updating detection signatures, or further investigating the identified threats.

4.3.3 Behavioral Analysis

Behavioral analysis is a crucial aspect of malware analysis that focuses on observing and understanding the actions and activities of a suspicious file or program. By analyzing the behavior, analysts can gain insights into the intentions and potential impact of the malware. In the context of Cuckoo, behavioral analysis is performed during the execution of the suspicious file in the instrumented guest virtual machine. Here's an overview of how behavioral analysis is conducted in Cuckoo:

1. Execution Monitoring:

- Cuckoo monitors the execution of the suspicious file in the guest virtual machine.
- It captures and records various activities and events generated by the file.

2. File System Activity:

- Cuckoo tracks the file system interactions initiated by the suspicious file.
- It logs file creations, modifications, deletions, and other file operations.
- Analysts can examine these activities to identify potential malicious actions, such as dropping or modifying files, creating persistence mechanisms, or encrypting data.

3. Registry Activity:

- Cuckoo monitors changes made by the suspicious file to the Windows Registry.
- It captures registry key creations, modifications, and deletions.
- Registry analysis helps identify malicious modifications, such as creating autorun entries, altering security settings, or modifying startup configurations.

4. Process Activity:

- Cuckoo records information about processes spawned by the suspicious file.
- It captures details such as process creation, termination, and inter-process communication.
- Process analysis helps identify any malicious processes spawned by the malware, such as those involved in lateral movement, privilege escalation, or persistence.

5. Network Communication:

- Cuckoo monitors network traffic generated by the suspicious file.
- It captures information about outgoing network connections, including IP addresses, ports, protocols, and payloads.
- Network analysis helps identify communication with malicious domains, command-and-control servers, or data exfiltration activities.

6. System Calls:

- Cuckoo intercepts and logs system calls made by the suspicious file during execution.
- System call analysis provides insights into low-level interactions with the operating system.
- It helps identify potentially malicious behavior, such as attempts to inject code, modify system configurations, or evade detection.

7. User Activity:

- Cuckoo can also monitor user-level activities during the execution of the suspicious file.
- It captures user input, interactions with the graphical user interface, and other user-related events.

- User activity analysis aids in understanding any attempts to trick or deceive the user, such as fake pop-ups or phishing attempts.

8. Behavioral Signatures:

- Cuckoo applies behavioral signatures or heuristics to detect known or suspicious patterns of behavior.
- These signatures can help identify common malware behaviors, such as file encryption, keylogging, or network scanning.

4.3.4 Network Monitoring

Network monitoring is an important component of malware analysis, and Cuckoo provides the capability to capture and analyze network traffic generated by the suspicious file or program. By monitoring network activity, analysts can gain insights into the communication patterns, command-and-control mechanisms, and potential data exfiltration performed by the malware. Here's an overview of network monitoring in Cuckoo:

1. Network Traffic Capture:

- Cuckoo integrates with network monitoring tools such as Wireshark or Suricata to capture network traffic.
- These tools are typically installed and configured to run alongside the Cuckoo analysis environment.

2. Virtual Network Setup:

- Cuckoo configures the virtual network environment to capture network traffic.
- This involves configuring network interfaces, routing, and capturing mechanisms within the guest virtual machine.

3. Network Traffic Analysis:

- During the execution of the suspicious file, Cuckoo captures and records network traffic generated by the malware.
- It logs network connections, including IP addresses, ports, protocols, and payloads.
- Cuckoo captures both inbound and outbound network activity to provide a comprehensive view of the malware's communication behavior.

4. Protocol Analysis:

- Cuckoo analyzes the captured network traffic to understand the protocols and communication patterns used by the malware.
- It decodes and interprets the network protocols involved in the communication, such as HTTP, DNS, SMTP, or IRC.
- Protocol analysis helps identify specific actions performed over the network, such as command-and-control communication, data exfiltration, or download of additional payloads.

5. DNS Analysis:

- Cuckoo examines DNS traffic to identify any malicious domain resolutions or suspicious DNS queries made by the malware.
- It can detect domain generation algorithms (DGAs) or DNS tunneling techniques employed by the malware.

6. Network Indicator Extraction:

- Cuckoo extracts network indicators from the captured traffic, such as URLs, IP addresses, domain names, or HTTP headers.
- These indicators can be used to identify known malicious infrastructure or for further investigation.

7. Network Signature Matching:

- Cuckoo can utilize signature-based detection mechanisms, such as Yara rules or Suricata rules, to identify network-based indicators of compromise (IOCs).
- By matching network signatures against the captured traffic, Cuckoo can flag potentially malicious patterns or behaviors.

8. Reporting:

- Cuckoo includes the network monitoring results in the analysis reports.
- The reports provide details about the observed network activity, including the communication flow, detected IOCs, and any suspicious or malicious connections.

4.3.5 Post-Analysis Activities

After completing the analysis of a suspicious file or program using Cuckoo, there are several post-analysis activities that analysts typically perform to further investigate the findings, mitigate the identified threats, and improve the overall security posture. Here are some common post-analysis activities:

1. Analysis Review:

- Analysts thoroughly review the generated analysis reports to gain a comprehensive understanding of the malware's behavior, capabilities, and potential impact.

- They analyze the captured data, behavioral indicators, network activity, and any detected malicious patterns to identify the key findings.

2. Threat Intelligence:

- Analysts leverage threat intelligence sources and databases to gather additional information about the identified indicators of compromise (IOCs), such as IP addresses, domains, file hashes, or malware families.

- This helps in understanding the broader threat landscape, attribution, and potential connections to known threat actors or campaigns.

3. Incident Response:

- If the analyzed malware is part of a larger security incident, analysts initiate incident response activities to contain and remediate the affected systems.

- They coordinate with incident response teams, stakeholders, and IT personnel to implement appropriate mitigation measures and restore the compromised systems.

4. Indicators Sharing:

- Analysts share the identified IOCs and other relevant indicators with internal security teams, industry peers, or trusted information sharing communities.

- This helps to improve collective defenses, enhance detection capabilities, and protect others from similar threats.

5. Malware Signature Updating:

- If the analyzed malware is previously unknown or undetected, analysts collaborate with antivirus vendors and security product providers to update their detection signatures.

- This ensures that the malware is recognized and blocked by security solutions in the future.

6. Security Controls Enhancement:

- Analysts use the insights gained from the analysis to strengthen security controls and preventive measures within their organization.

- They update firewall rules, intrusion detection/prevention systems, antivirus policies, or web filtering to better defend against similar threats.

7. System Patching and Vulnerability Management:

- Analysts assess if the analyzed malware exploited any known vulnerabilities or weaknesses in the system.

- They coordinate with system administrators and IT teams to apply necessary patches, updates, or configuration changes to address these vulnerabilities.

8. Lessons Learned and Documentation:

- Analysts document the analysis process, key findings, lessons learned, and any new techniques or tools utilized during the investigation.
- This documentation serves as a knowledge base for future reference, training, and improving the effectiveness of the malware analysis process.

9. Continuous Improvement:

- Analysts continuously evaluate and enhance the malware analysis workflow, methodologies, and tools used.
- They stay updated with the latest trends, techniques, and emerging threats in the field of cybersecurity to adapt their practices accordingly.

4.4 Installing and Setting-up

In this section, we will install and set up Cuckoo and its prerequisites. Analyze the reports and step-by-step guide to install and find malware analysis by Cuckoo.

4.4.1 Dual Booting

Dual booting Windows and Ubuntu allows you to have both operating systems installed on the same computer, giving you the flexibility to choose between them at startup. Here's a general overview of the steps involved in setting up a dual boot configuration with Windows and Ubuntu:

1. Backup your data:

- Before making any changes to your computer's disk partitions, it's crucial to back up any important data to ensure it's not lost during the installation process.

2. Prepare installation media:

- Download the latest versions of Windows and Ubuntu ISO files from their respective official websites.
- Create bootable USB drives for both operating systems using tools like Rufus or UNetbootin.

3. Partition your hard drive:

- In Windows, use the Disk Management utility to shrink your existing Windows partition and create free space for installing Ubuntu.
- Allocate sufficient space for Ubuntu while leaving the Windows partition intact.

4. Install Windows:

- Insert the Windows installation USB and restart your computer.
- Follow the on-screen instructions to install Windows in the partition you created.

5. Install Ubuntu:

- Insert the Ubuntu installation USB and restart your computer.
- During the Ubuntu installation process, choose the option to install alongside Windows.
- Select the free space you allocated earlier for Ubuntu and follow the on-screen instructions to complete the installation.

6. Configure the boot loader:

- By default, Ubuntu's bootloader (GRUB) should detect both Windows and Ubuntu installations and present a menu at startup for choosing the operating system.
- You can select the default operating system and customize the bootloader settings if desired.

7. Test the dual boot:

- Restart your computer and verify that the boot menu appears, allowing you to choose between Windows and Ubuntu.
- Test booting into both operating systems to ensure they function correctly.

8. Update and maintain your dual boot setup:

- Regularly update both Windows and Ubuntu to ensure you have the latest security patches and software updates.
- Take precautions when modifying disk partitions or performing major system changes to avoid any issues with the dual boot configuration.

4.4.2 Host Preparation

Before setting up Cuckoo, there are several host preparation steps you need to take to ensure your system is properly configured and ready for malware analysis. Here's an overview of the host preparation process for setting up Cuckoo:

1. Operating System:

- Choose a suitable host operating system (OS) that is supported by Cuckoo.

- Cuckoo supports various Linux distributions, such as Ubuntu, Debian, CentOS, or Fedora.
- Ensure that your chosen OS is up to date with the latest security patches and updates.

2. Virtualization Software:

- Install a virtualization software on the host machine.
- Cuckoo primarily supports two virtualization options: VirtualBox and VMware.
- Install the preferred virtualization software and make sure it's configured correctly.

3. Hardware Virtualization Support:

- Enable hardware virtualization support in your computer's BIOS settings.
- This feature is often referred to as Intel VT-x (for Intel processors) or AMD-V (for AMD processors).
- Enabling hardware virtualization ensures optimal performance and compatibility for virtualization-based malware analysis.

4. Networking Setup:

- Configure the network interface on the host machine.
- Cuckoo requires network connectivity for communication between the host and the virtual machines.
- Set up the network interface with appropriate IP addressing and ensure it has internet connectivity.

5. Software Dependencies:

- Install the necessary software dependencies required by Cuckoo.
- These dependencies include Python, pip (Python package manager), and additional system libraries and tools.
- Refer to the Cuckoo documentation or setup guides for the specific dependencies and their installation instructions.

6. Python Virtual Environment:

- Set up a Python virtual environment for Cuckoo.
- Creating a virtual environment helps isolate Cuckoo's dependencies and ensures a clean and controlled environment for its execution.
- Install and activate the Python virtual environment.

7. Cuckoo Installation:

- Download the latest version of Cuckoo from the official Cuckoo Sandbox GitHub repository.

- Extract the Cuckoo files and configure the necessary settings, such as the virtualization software, networking options, and analysis settings.
- Follow the installation instructions provided in the Cuckoo documentation or setup guides.

8. Security Measures:

- Implement necessary security measures to protect the host machine and the Cuckoo setup.
- Apply appropriate firewall rules, isolate the Cuckoo environment from the rest of the network, and follow best practices for system hardening and access control.

9. Testing and Validation:

- Test the Cuckoo setup by running a sample analysis and verifying that it functions as expected.
- Monitor the analysis process, review the generated reports, and ensure that the virtualized environment is working properly.

4.4.3 Guest Preparation

In addition to the host preparation steps, there are certain guest preparation steps you need to take to properly configure the guest virtual machine (VM) for malware analysis with Cuckoo. The guest VM is where the suspicious files or programs will be executed and analyzed. Here's an overview of the guest preparation process:

1. Choose a Guest Operating System:

- Select the appropriate guest operating system to be installed within the virtual machine.
- Cuckoo supports various guest operating systems, including different versions of Windows, Linux, and macOS.

2. Install the Guest Operating System:

- Create a new virtual machine within the virtualization software (e.g., VirtualBox or VMware).
- Install the chosen guest operating system on the virtual machine using the installation media (ISO or installation disc).

3. Update the Guest Operating System:

- After installing the guest operating system, apply all available updates and security patches.
- This helps ensure that the guest operating system is up to date with the latest fixes and vulnerabilities.

4. Install Guest Additions/Tools:

- Install the appropriate guest additions or tools provided by the virtualization software.
- These tools enhance the integration between the host and guest systems, improving performance, screen resolution, shared folders, and clipboard functionality.

5. Disable or Uninstall Unnecessary Software:

- Remove any unnecessary or potentially conflicting software from the guest operating system.
- Disable or uninstall applications, services, or features that may interfere with the malware analysis process or impact system stability.

6. Configure Network Settings:

- Adjust the network settings of the guest VM to ensure proper network connectivity.
- Set the network adapter to bridge mode or NAT mode, depending on the desired network configuration for analysis.

7. Create a Snapshot (Optional):

- Optionally, create a snapshot of the clean guest VM configuration before conducting any malware analysis.
- Snapshots allow you to revert the virtual machine to its initial clean state, making it easier to start fresh analyses.

8. Install Required Analysis Tools:

- Install any additional analysis tools or software that may be required for the specific malware analysis scenarios.
- Examples include debuggers, packet capture tools, monitoring agents, or specific analysis tools for forensic examination.

9. Enable Remote Access (Optional):

- If desired, configure the guest VM to allow remote access for monitoring and controlling the analysis process.
- This can be done by enabling remote desktop (RDP) or setting up SSH access.

4.4.4 Cuckoo INIT & Routing

Cuckoo INIT and routing are two important components of the Cuckoo sandbox setup that help in configuring the network environment for malware analysis. Here's an explanation of Cuckoo INIT and routing in the context of Cuckoo sandbox:

1. Cuckoo INIT:

- Cuckoo INIT is a component of Cuckoo sandbox that initializes the network environment for the analysis of malware samples.

- It sets up the necessary network interfaces, routing rules, and network configurations within the virtualized environment to enable network communication.

2. Network Interfaces:

- Cuckoo INIT configures virtual network interfaces within the guest virtual machine (VM) to enable network connectivity.

- It may create multiple network interfaces, such as a bridged interface or NAT interface, depending on the network configuration chosen during the Cuckoo setup.

3. Routing:

- Routing in the context of Cuckoo refers to the configuration of network routing rules within the guest VM to redirect network traffic to Cuckoo's monitoring components.

- Cuckoo utilizes various routing techniques to intercept and capture network traffic generated by the malware sample being analyzed.

4. Port Forwarding:

- Cuckoo can set up port forwarding rules to redirect network traffic from specific ports or protocols to the Cuckoo monitoring components.

- This allows Cuckoo to capture and analyze network activity, including HTTP requests, DNS queries, or other communication protocols used by the malware.

5. Monitoring and Capturing:

- Once the network environment is set up by Cuckoo INIT and the routing rules are configured, Cuckoo can monitor and capture the network traffic generated by the malware.

- It captures network packets, logs network connections, and records the behavior of the malware's communication with external entities.

6. Network Analysis:

- Cuckoo analyzes the captured network traffic to extract relevant information and behavior patterns exhibited by the malware.

- It can decode and interpret network protocols, identify communication with command-and-control servers, detect data exfiltration, or analyze the content of network payloads.

4.4.5 Report and Web Interface

Cuckoo Sandbox provides a web interface and reporting capabilities to help users analyze and understand the behavior and impact of the analyzed malware. Here's an overview of the report generation and web interface features in Cuckoo:

1. Report Generation:

- After the analysis of a malware sample is completed, Cuckoo generates a comprehensive analysis report that summarizes the findings and observed behavior.
- The report includes information such as system changes, network activity, file modifications, registry changes, process activity, and any other relevant behavior exhibited by the malware.
- Reports can be generated in various formats, including HTML, JSON, and PDF, depending on the configuration and user preferences.

2. Web Interface:

- Cuckoo Sandbox provides a web interface that allows users to access and interact with the analysis reports and other analysis-related information.
- The web interface provides a graphical user interface (GUI) to browse through the reports, view analysis details, and access additional features and functionalities.

3. Report Visualization:

- The web interface in Cuckoo includes visualization features that help users interpret the analysis results more effectively.
- Graphical representations, charts, and diagrams are used to display the network traffic, process activity, file activity, and other analyzed data in a more understandable format.

4. Report Search and Filtering:

- The web interface allows users to search for specific reports based on various criteria, such as file names, analysis dates, or indicators of compromise (IOCs).
- Filtering options are available to narrow down the reports based on specific attributes or behavior exhibited by the malware.

5. IOC Extraction and Highlighting:

- Cuckoo's web interface can extract indicators of compromise (IOCs) from the analyzed malware and highlight them in the reports.
- IOCs may include IP addresses, domains, file hashes, URLs, registry keys, or other identifiable patterns that can help in threat intelligence and detection.

6. User Interaction:

- The web interface provides options for users to comment on reports, tag them with relevant labels, or collaborate with other analysts.
- Users can add their own observations, notes, or insights to the analysis reports to enhance the collective knowledge and understanding of the malware.

7. Integration with External Tools:

- Cuckoo's web interface can integrate with external security tools or services to provide additional context and enrich the analysis reports.
- Integration with threat intelligence platforms, antivirus scanners, or other analysis tools can enhance the accuracy and depth of the analysis results.

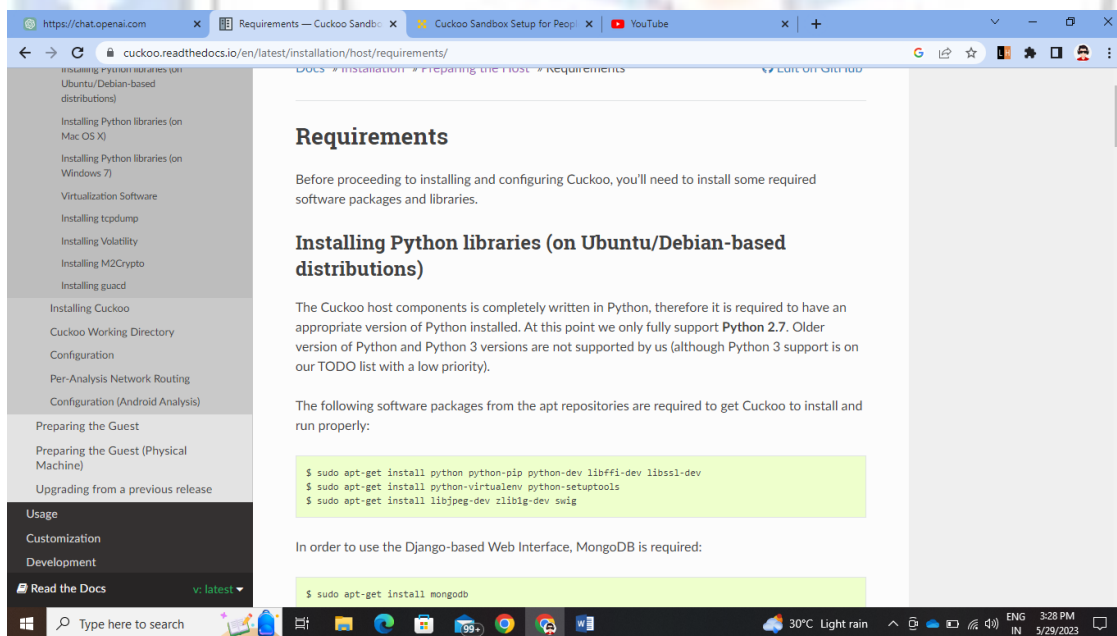
4.4.6 Procedure

1. First do the Dual Booting Windows and Ubuntu 16.04 OS by booting up in pen drive and then BIOS environment.

Do refer to this video:

<https://youtu.be/QKn5U2esuRk>

2. After booting up OS then follow the Installation Guide Prepare the Host with requirements



3. Install Mongo DB, M2crypto, TCP dump, Guacd, and Virtualization software as VirtualBox 5.2

https://chat.openai.com x Requirements — Cuckoo Sandbo x Cuckoo Sandbox Setup for Peop x YouTube

cuckoo.readthedocs.io/en/latest/installation/host/requirements/

Installing M2Crypto

Currently the `M2Crypto` library is only supported when `SWIG` has been installed. On Ubuntu/Debian-like systems this may be done as follows:

```
$ sudo apt-get install swig
```

If `SWIG` is present on the system one may install `M2Crypto` as follows:

```
$ sudo pip install m2crypto==0.24.0
```

Installing guacd

`guacd` is an optional service that provides the translation layer for RDP, VNC, and SSH for the remote control functionality in the Cuckoo web interface.

Without it, remote control won't work. Versions 0.9.9 and up will work, but we recommend installing the latest version. On an Ubuntu 17.04 machine the following command will install version `0.9.9-2`:

```
$ sudo apt install libguac-client-rdp0 libguac-client-vnc0 libguac-client-ssh0 guacd
```

If you only want RDP support you can skip the installation of the `libguac-client-vnc0` and `libguac-client-ssh0` packages

Introduction

- Installation
 - Preparing the Host
 - Requirements
 - Installing Python libraries (on Ubuntu/Debian-based distributions)
 - Installing Python libraries (on Mac OS X)
 - Installing Python libraries (on Windows 7)
 - Virtualization Software
 - Installing tcpdump
 - Installing Volatility
 - Installing M2Crypto
 - Installing guacd
 - Installing Cuckoo
 - Cuckoo Working Directory
 - Configuration
 - Per-Analysis Network Routing
 - Configuration (Android Analysis)
- Preparing the Guest
- Preparing the Guest (Physical Machine)

Read the Docs v: latest

Type here to search

30°C Light rain 3:34 PM 5/29/2023

4. Create Cuckoo User

https://chat.openai.com x Installing Cuckoo — Cuckoo San x Cuckoo Sandbox Setup for Peop x YouTube

cuckoo.readthedocs.io/en/latest/installation/host/installation/

Create a user

You can either run Cuckoo from your own user or create a new one dedicated just for your sandbox setup. Make sure that the user that runs Cuckoo is the same user that you will use to create and run the virtual machines (at least in the case of VirtualBox), otherwise Cuckoo won't be able to identify and launch these Virtual Machines.

Create a new user:

```
$ sudo adduser cuckoo
```

If you're using VirtualBox, make sure the new user belongs to the "vboxusers" group (or the group you used to run VirtualBox):

```
$ sudo usermod -a -G vboxusers cuckoo
```

If you're using KVM or any other libvirt based module, make sure the new user belongs to the "libvirt" group (or the group your Linux distribution uses to run libvirt):

```
$ sudo usermod -a -G libvirt cuckoo
```

Raising file limits

As outlined in the FAQ entry `IOError: [Errno 24] Too many open files` one may want to bump the file count limits before starting Cuckoo as otherwise some samples will fail to properly process the

latest

Search docs

FAQ

Introduction

- Installation
 - Preparing the Host
 - Requirements
 - Installing Cuckoo
 - Create a user
 - Raising file limits
 - Install Cuckoo
 - Install Cuckoo from file
 - Build/Install Cuckoo from source
 - Cuckoo Working Directory
 - Configuration
 - Per-Analysis Network Routing
 - Configuration (Android Analysis)
 - Preparing the Guest
 - Preparing the Guest (Physical Machine)
- Read the Docs v: latest

Type here to search

30°C Light rain 3:35 PM 5/29/2023

dabeaz

Tough Courses for Programmers. Sure, you've taken an elevator, but could you code an elevator?

Ads by EthicalAds

5. Install Virtualenv and in that venv install cuckoo



Installation

- Preparing the Host
 - Requirements
 - Installing Cuckoo
 - Create a user
 - Raising file limits
 - Install Cuckoo
 - Install Cuckoo from file
 - Build/Install Cuckoo from source
 - Cuckoo Working Directory
 - Configuration
 - Per-Analysis Network Routing
 - Configuration (Android Analysis)
- Preparing the Guest
 - Preparing the Guest (Physical Machine)
 - Upgrading from a previous release
- Usage
- Customization
- Development
- Final Remarks
- Read the Docs v: latest

Warning

It is not unlikely that you'll be missing one or more system packages required to build various Python dependencies. Please read and re-read [Requirements](#) to resolve these sorts of issues.

```
$ sudo pip install -U pip setuptools
$ sudo pip install -U cuckoo
```

Although the above, a *global* installation of Cuckoo in your OS works mostly fine, we **highly** recommend installing Cuckoo in a `virtualenv`, which looks roughly as follows:

```
$ virtualenv venv
$ . venv/bin/activate
(venv)$ pip install -U pip setuptools
(venv)$ pip install -U cuckoo
```

Note

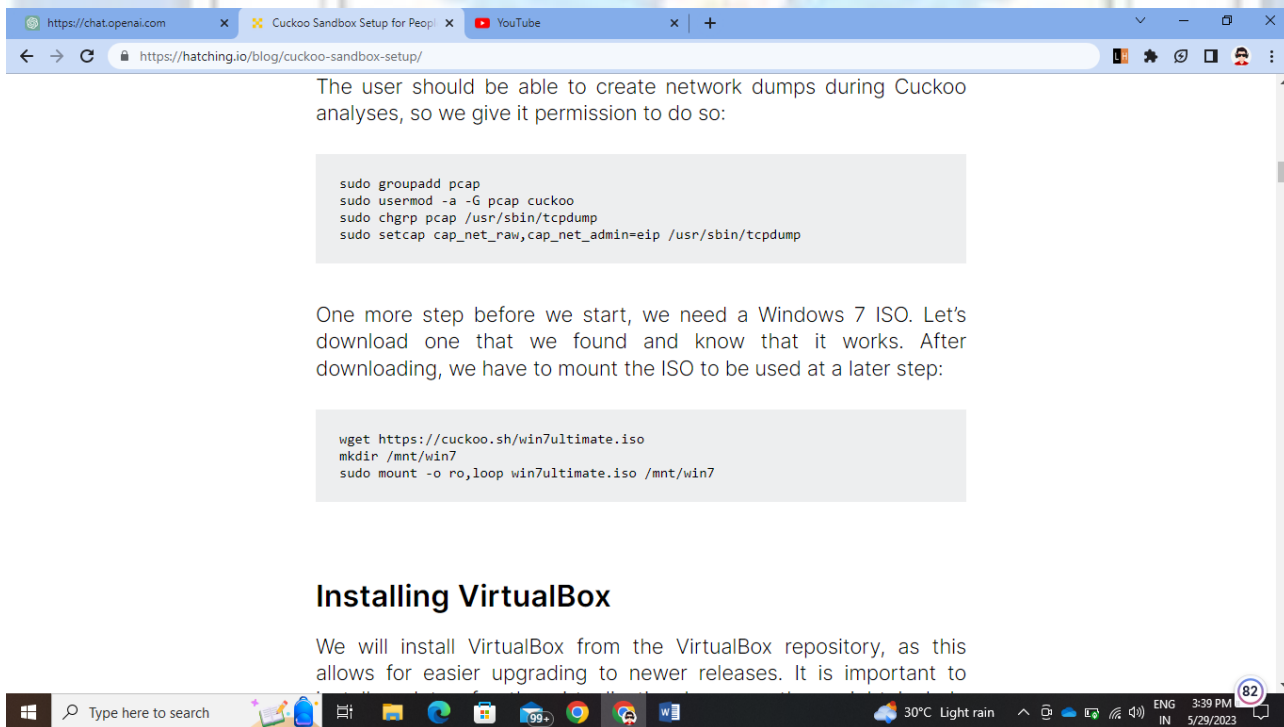
Depending on how you have set up your environment (virtualenvs etc.) you may need to specify the version of `pip` to use. Just replace `pip` in the commands above with `pip2`.

Some reasons for using a `virtualenv`:

- Cuckoo's dependencies may not be entirely up-to-date, but instead pin to a known-to-work-properly version.
- The dependencies of other software installed on your system may conflict with those required

6. Now Prepare the Guest Part of Cuckoo Install ISO of windows7 and follow the blog below

<https://hatching.io/blog/cuckoo-sandbox-setup/>



The user should be able to create network dumps during Cuckoo analyses, so we give it permission to do so:

```
sudo groupadd pcap
sudo usermod -a -G pcap cuckoo
sudo chgrp pcap /usr/sbin/tcpdump
sudo setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump
```

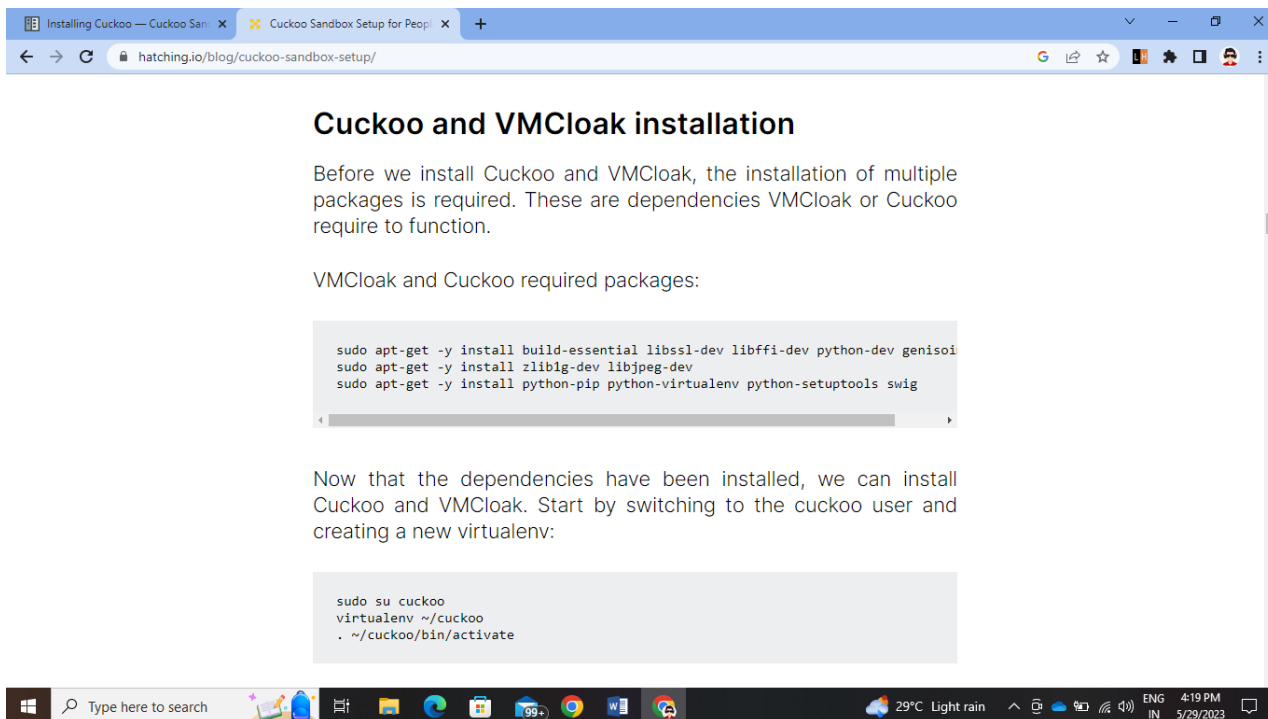
One more step before we start, we need a Windows 7 ISO. Let's download one that we found and know that it works. After downloading, we have to mount the ISO to be used at a later step:

```
wget https://cuckoo.sh/win7ultimate.iso
mkdir /mnt/win7
sudo mount -o ro,loop win7ultimate.iso /mnt/win7
```

Installing VirtualBox

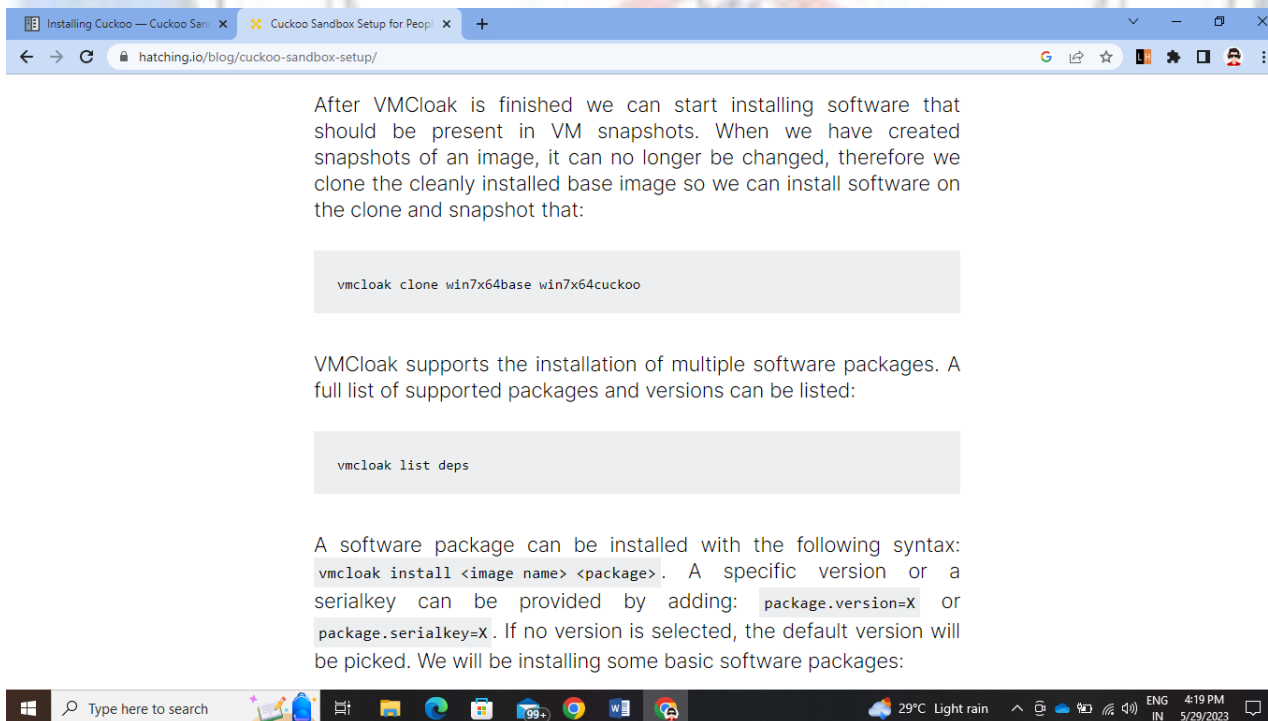
We will install VirtualBox from the VirtualBox repository, as this allows for easier upgrading to newer releases. It is important to

7. Install Vmcloak and Virtualenv and activate windows



8. Make VMs as per your processor and ram size as this syntax. Then clone the windows on the vmcloak and install all dependencies

```
vmcloak init --verbose --win7x64 win7x64base --cpus 2 --ramsize 1036 -iso-mount /mnt/win7
```



9. Install Snapshots and list all vms in the terminal of Ubuntu

Installing Cuckoo — Cuckoo Sandbox Setup for People

hatching.io/blog/cuckoo-sandbox-setup/

snapshot created. After snapshotting, it is no longer possible to change the image. The syntax of the snapshot command is:

```
vmcloak snapshot <options> <image name> <vmname> <ip to use>
```

Using the `--count` parameter, we can create multiple snapshots at once. Let's create four:

```
vmcloak snapshot --count 4 win7x64cuckoo 192.168.56.101
```

This command will create VMs `win7x64cuckoo1-4` with IPs `192.168.56.101-104`.

After VMCloak is finished, the VMs can be listed using:

```
vmcloak list vms
```

Activate Windows
Go to Settings to activate Windows.

Configuring Cuckoo

Type here to search

Near record

ENG IN 4:20 PM 5/29/2023

10. Add vms to the machine and delete the cuckoo1 machine data from `./cuckoo/conf/virtualbox.py` by nano or vim command

Installing Cuckoo — Cuckoo Sandbox Setup for People

hatching.io/blog/cuckoo-sandbox-setup/

Adding VMs

We are using VirtualBox in our setup, this is the default [Machinery module](#) that Cuckoo uses. We have to remove some default settings from its configuration file `virtualbox.conf`.

All Cuckoo configuration files can be found at `$CWD/conf/`. Open `$CWD/conf/virtualbox.conf` and remove the entries in the `machines = cuckoo1` line.

Time to add the created VMs to Cuckoo. We will use the `cuckoo machine --add <vm name> <ip>` to tell Cuckoo to add the machine to its configuration. This has to be done for each machine, so let's make life easier and use `vmcloak list vms`:

```
while read -r vm ip; do cuckoo machine --add $vm $ip; done < <(vmcloak list vms)
```

To install the Cuckoo signatures and latest monitor, we run the following command:

Activate Windows
Go to Settings to activate Windows.

Earnings upcoming

ENG IN 4:23 PM 5/29/2023

11. Do the network configuration host and vm server with this command and put wps130 instead of eth0 as server can be known by ifconfig on the terminal.

```
(cuckoo) cuckoo@param-HP-Notebook:~$ sudo sysctl -w net.ipv4.conf.vboxnet0.forwarding=1
[sudo] password for cuckoo:
sysctl: cannot stat /proc/sys/net/ipv4/conf/vboxnet0/forwarding: No such file or directory
(cuckoo) cuckoo@param-HP-Notebook:~$ vmcloak-vboxnet0
(cuckoo) cuckoo@param-HP-Notebook:~$ sudo sysctl -w net.ipv4.conf.vboxnet0.forwarding=1
net.ipv4.conf.vboxnet0.forwarding = 1
(cuckoo) cuckoo@param-HP-Notebook:~$ ifconfig
enp7s0    Link encap:Ethernet  HWaddr 30:e1:71:11:97:8c
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:805 errors:0 dropped:0 overruns:0 frame:0
          TX packets:805 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:105962 (105.9 KB)  TX bytes:105962 (105.9 KB)

vboxnet0  Link encap:Ethernet  HWaddr 0a:00:27:00:00:00
          inet addr:192.168.56.1  Bcast:192.168.56.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

virbr0    Link encap:Ethernet  HWaddr 00:00:00:00:00:00
          inet addr:192.168.122.1  Bcast:192.168.122.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wlp13s0   Link encap:Ethernet  HWaddr 94:e9:79:dc:6a:f7
          inet addr:192.168.43.62  Bcast:192.168.43.255  Mask:255.255.255.0
          inet6 addr: fe80::f0:2dbf:ef26:b5c4/64 Scope:Link
          inet6 addr: 2405:204:10a3:8751:e045:316f:7d6f:7d0c/64 Scope:Global
```

12. Now open the terminator and make four parts. Run the router command on one phase and run web host server command and cuckoo -debug command on third pannel

```
cuckoo@param-HP-Notebook: ~ 80x23
2023-05-29 06:40:32,564 [cuckoo.apps.rooter] INFO: Processing command: forward_d
isable vboxnet0 wlp13s0 192.168.56.101
2023-05-29 06:40:32,574 [cuckoo.apps.rooter] INFO: Processing command: forward_d
isable vboxnet0 wlp13s0 192.168.56.102
2023-05-29 06:40:32,589 [cuckoo.apps.rooter] INFO: Processing command: forward_d
isable vboxnet0 wlp13s0 192.168.56.103
2023-05-29 06:40:32,601 [cuckoo.apps.rooter] INFO: Processing command: forward_d
isable vboxnet0 wlp13s0 192.168.56.104
2023-05-29 06:42:01,804 [cuckoo.apps.rooter] INFO: Processing command: nic_avail
able wlp13s0
2023-05-29 06:42:01,999 [cuckoo.apps.rooter] INFO: Processing command: drop_enab
le 192.168.56.101 192.168.56.1 2042
2023-05-29 06:42:02,203 [cuckoo.apps.rooter] INFO: Processing command: forward_e
nable vboxnet0 wlp13s0 192.168.56.101
2023-05-29 06:42:02,383 [cuckoo.apps.rooter] INFO: Processing command: srcroute_
enable main 192.168.56.101
2023-05-29 06:48:03,751 [cuckoo.apps.rooter] INFO: Processing command: forward_d
isable vboxnet0 wlp13s0 192.168.56.101
2023-05-29 06:48:03,964 [cuckoo.apps.rooter] INFO: Processing command: srcroute_
disable main 192.168.56.101
2023-05-29 06:48:04,192 [cuckoo.apps.rooter] INFO: Processing command: drop_disa
ble 192.168.56.101 192.168.56.1 2042

cuckoo@param-HP-Notebook: ~ 80x23
2023-05-29 06:49:29,025 [cuckoo.core.plugins] DEBUG: Analysis matched signature:
antimv_network_adapters
2023-05-29 06:49:29,025 [cuckoo.core.plugins] DEBUG: Analysis matched signature:
pe_features
2023-05-29 06:49:29,025 [cuckoo.core.plugins] DEBUG: Analysis matched signature:
privilege_luid_check
2023-05-29 06:49:29,026 [cuckoo.core.plugins] DEBUG: Analysis matched signature:
memdump_urls
2023-05-29 06:49:29,027 [cuckoo.core.plugins] DEBUG: Analysis matched signature:
queries_programs
2023-05-29 06:49:29,028 [cuckoo.core.plugins] DEBUG: Analysis matched signature:
antimv_firmware
2023-05-29 06:49:29,028 [cuckoo.core.plugins] DEBUG: Analysis matched signature:
wmi_antimv
2023-05-29 06:49:42,051 [cuckoo.core.plugins] DEBUG: Executed reporting module "
JsonDump"
2023-05-29 06:50:07,874 [cuckoo.core.plugins] DEBUG: Executed reporting module "
MongoDB"
2023-05-29 06:50:07,874 [cuckoo.core.scheduler] INFO: Task #16: reports generati
on completed
2023-05-29 06:50:07,958 [cuckoo.core.scheduler] INFO: Task #16: analysis procedu
re completed

cuckoo@param-HP-Notebook: ~ 80x23
[29/May/2023 06:56:28] "GET /static/js/cuckoo/sticky.js HTTP/1.1" 304 0
[29/May/2023 06:56:28] "GET /static/js/cuckoo/loader.js HTTP/1.1" 304 0
[29/May/2023 06:56:28] "GET /static/js/cuckoo/analysis_feedback.js HTTP/1.1" 304
0
[29/May/2023 06:56:28] "GET /static/js/cuckoo/analysis_sidebar.js HTTP/1.1" 304
0
[29/May/2023 06:56:28] "GET /static/js/cuckoo/submission.js HTTP/1.1" 304 0
[29/May/2023 06:56:28] "GET /static/js/cuckoo/process_tree.js HTTP/1.1" 304 0
[29/May/2023 06:56:28] "GET /static/js/cuckoo/analysis_network.js HTTP/1.1" 304
0
[29/May/2023 06:56:28] "GET /static/js/cuckoo/recent.js HTTP/1.1" 304 0
[29/May/2023 06:56:28] "GET /static/js/cuckoo/app.js HTTP/1.1" 304 0
[29/May/2023 06:56:28] "GET /static/js/cuckoo/rdp.js HTTP/1.1" 304 0
[29/May/2023 06:56:28] "GET /static/graphic/cuckoo_inverse.png HTTP/1.1" 304 0
[29/May/2023 06:56:28] "GET /static/graphic/cuckoo-coffee-cup.png HTTP/1.1" 304
0
[29/May/2023 06:56:28] "GET /static/images/prev.png HTTP/1.1" 304 0
[29/May/2023 06:56:28] "GET /static/images/next.png HTTP/1.1" 304 0
[29/May/2023 06:56:28] "GET /static/images/close.png HTTP/1.1" 304 0
[29/May/2023 06:56:28] "GET /static/images/loading.gif HTTP/1.1" 304 0
[29/May/2023 06:56:28] "POST /analysis/api/tasks/recent/ HTTP/1.1" 200 3312
[29/May/2023 06:56:30] "POST /analysis/api/tasks/recent/ HTTP/1.1" 200 13
```

13. Enable the mongodb from no to yes in reporting.conf and initiate the command can follow this video for host and guest preparation:

- https://youtu.be/QLQS4gk_IFU
- <https://youtu.be/FsF56772ZvU>

Results will only show if they have been processed after MongoDB has been enabled. As Cuckoo will not store them in there by default.

We start by opening `$CWD/conf/reporting.conf` and find the `[MongoDB]` section. Change `enabled = no` to `enabled = yes`. No further configuration changes are required, unless your MongoDB setup requires a user, runs on a non-standard port, or runs remotely.

The built-in web server

This server should not be used for production environments. It is a development server. It can be used for small setups, but should not be exposed to the internet.

The server can be started by running:

```
cuckoo web --host 127.0.0.1 --port 8080
```

We can now submit tasks and view results in the web interface. Cuckoo must be running for analyses to start, otherwise tasks will remain on the *pending* status.

Activate Windows
Go to Settings to activate Windows.

14. See the virtual machine and power on the VM 101 and put agent folder in startup in C drive

File Machine Help

New Settings Discard Start

192.168.56.1011 (vmcloak) Powered Off

192.168.56.1012 (vmcloak) Saved

192.168.56.1013 (vmcloak) Saved

192.168.56.1014 (vmcloak) Saved

Machine Tools Global Tools

Welcome to VirtualBox!

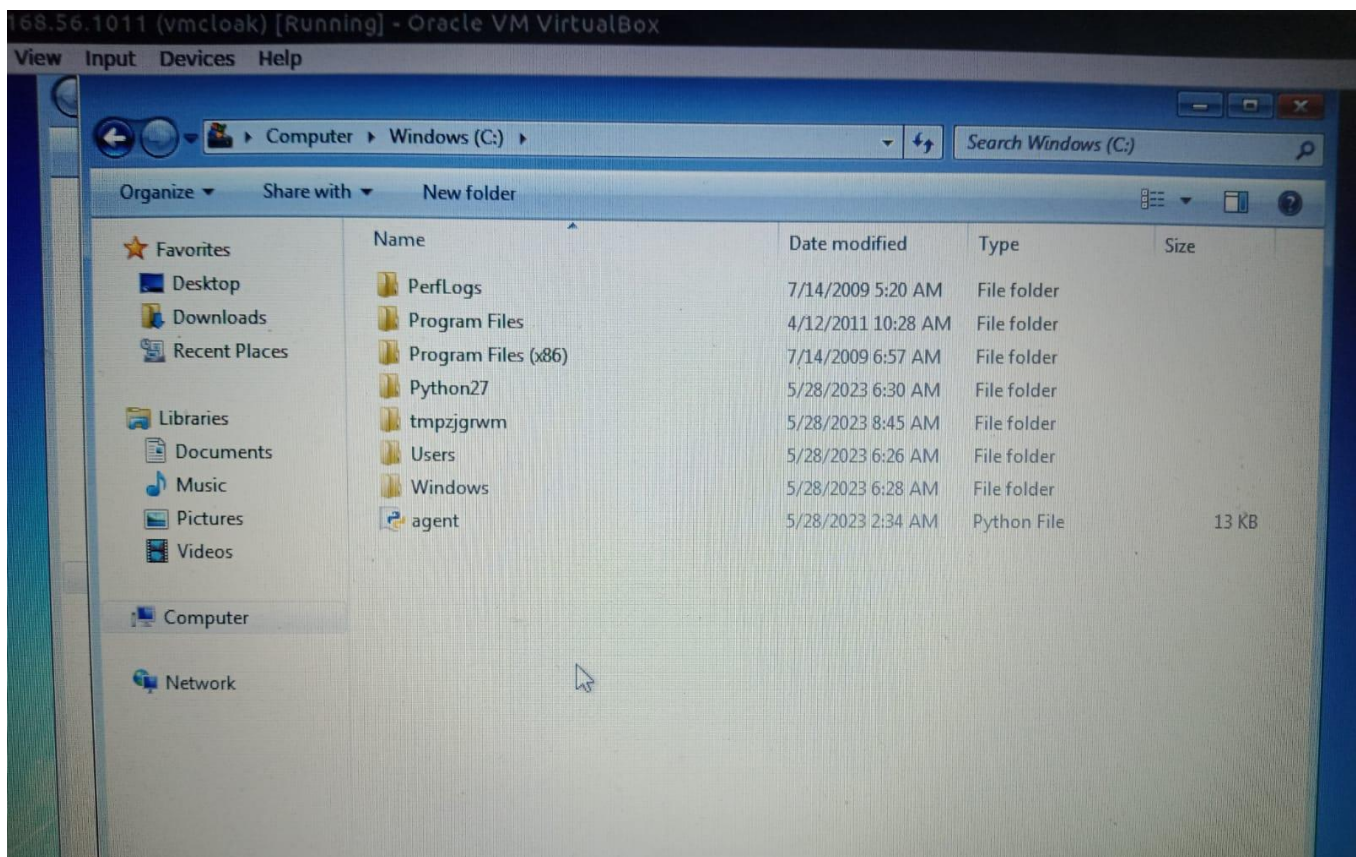
The left part of this window lists all virtual machines and virtual machine groups on your computer.

The right part of this window represents a set of tools which are currently opened (or can be opened) for the currently chosen machine. For a list of currently available tools check the corresponding menu at the right side of the main tool bar located at the top of the window. This list will be extended with new tools in future releases.

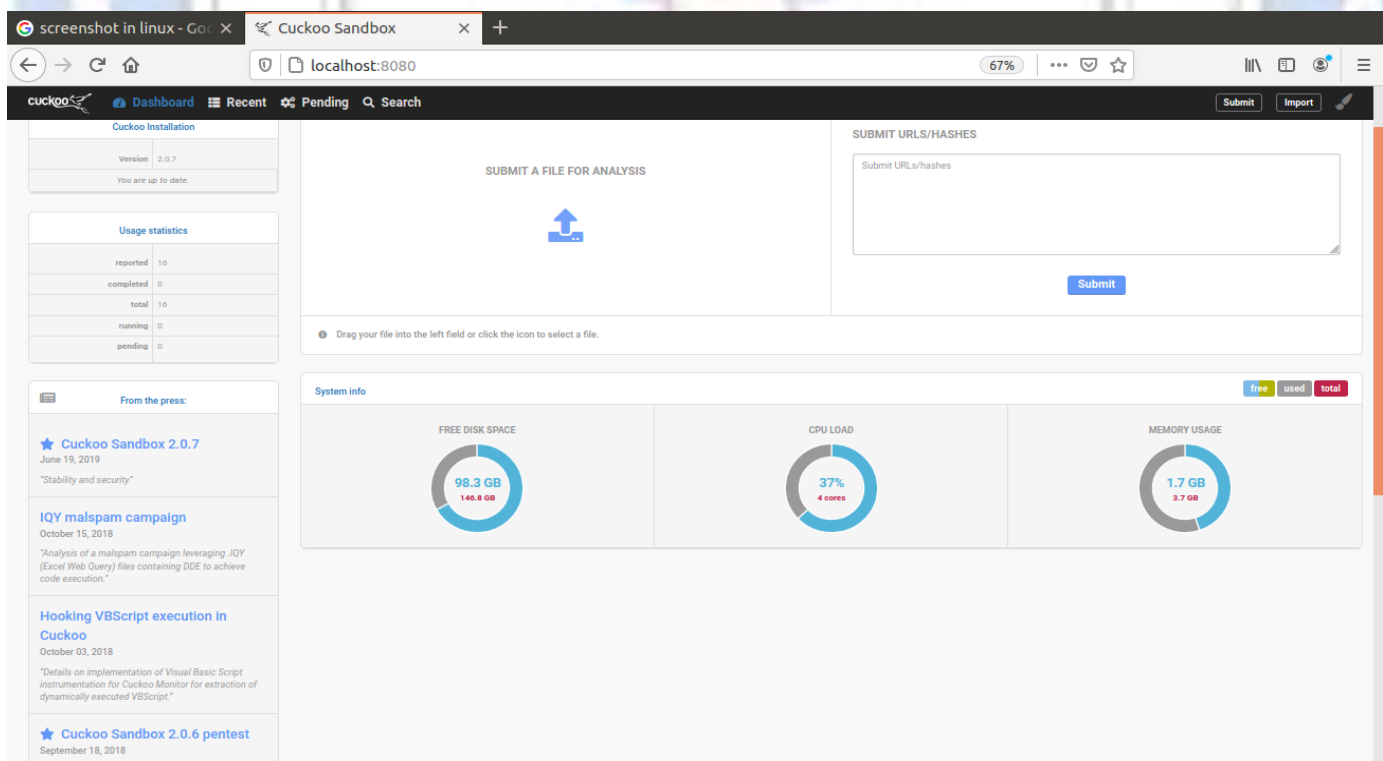
You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

Details
Tool to observe virtual machine (VM) details. Reflects groups of properties for the currently chosen VM and allows basic operations on certain properties (like the machine storage devices).

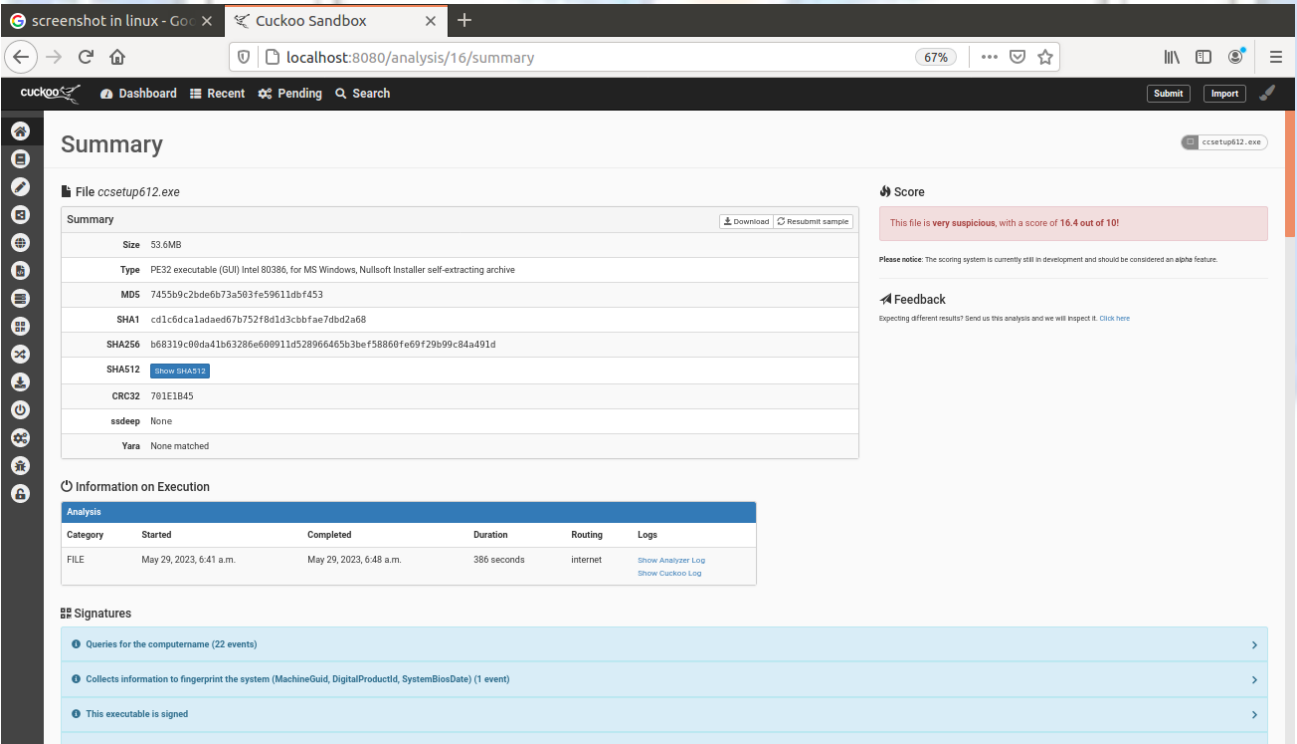
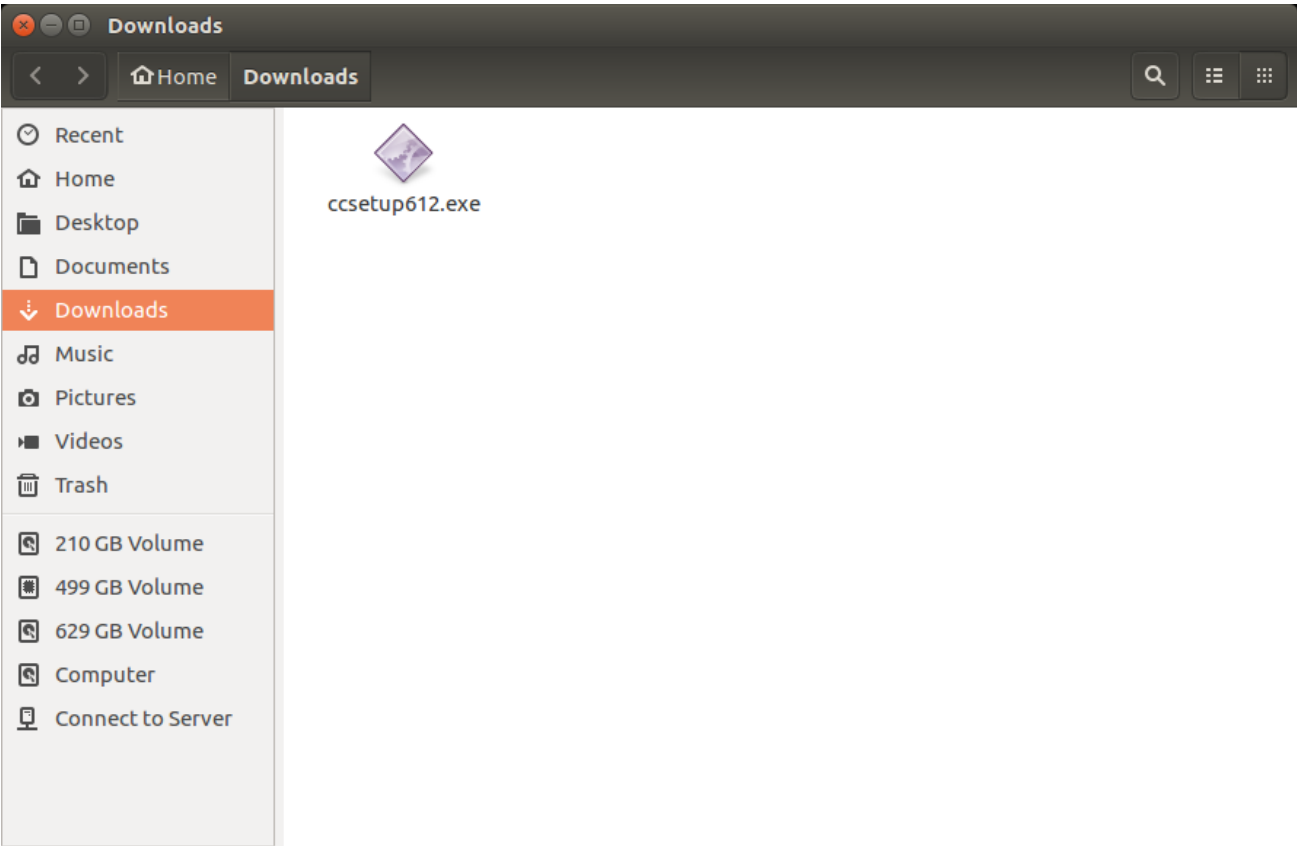
Snapshots
Tool to control virtual machine (VM) snapshots. Reflects snapshots created for the currently selected VM and allows snapshot operations like create, remove, restore (make current) and observe their properties. Allows to edit snapshot attributes like name and description.



15. Start the webhost by localhost:8080/ and put the url or file to test



16. We will test for CCleaner exe file that is malicious after submission it will open VM window7 and intsalll the exe file and then after some time generate the reports and score of malicious content in it.



screenshot in linux - Go... x Cuckoo Sandbox x +

localhost:8080/analysis/16/summary

67%

Submit Import

Signatures

- Queries for the computername (22 events)
- Collects information to fingerprint the system (MachineGuid, DigitalProductId, SystemBiosDate) (1 event)
- This executable is signed
- Checks amount of memory in system, this can be used to detect virtual machines that have a low amount of memory available (1 event)
- The executable contains unknown PE section names indicative of a packer (could be a false positive) (1 event)
- One or more potentially interesting buffers were extracted, these generally contain injected code, configuration data, etc.
- HTTP traffic contains suspicious features which may be indicative of malware related traffic (1 event)
- Performs some HTTP requests (17 events)
- Sends data using the HTTP POST Method (1 event)
- Allocates read-write-execute memory (usually to unpack itself) (44 events)
- Checks whether any human activity is being performed by constantly checking whether the foreground window changed
- A process attempted to delay the analysis task. (1 event)
- Queries the disk size which could be used to detect virtual machine with small fixed size or dynamic allocation (6 events)
- Steals private information from local Internet browsers (5 events)
- Creates executable files on the filesystem (2 events)
- Creates a shortcut to an executable file (15 events)

localhost:8080/analysis/16/summary#signature_allocates_rwx

17. We can start again cuckoo with these commands

cuckoo setup.odt - LibreOffice Writer

```
sudo su cuckoo

. ~/cuckoo/bin/activate

cuckoo rooter --sudo --group cuckoo

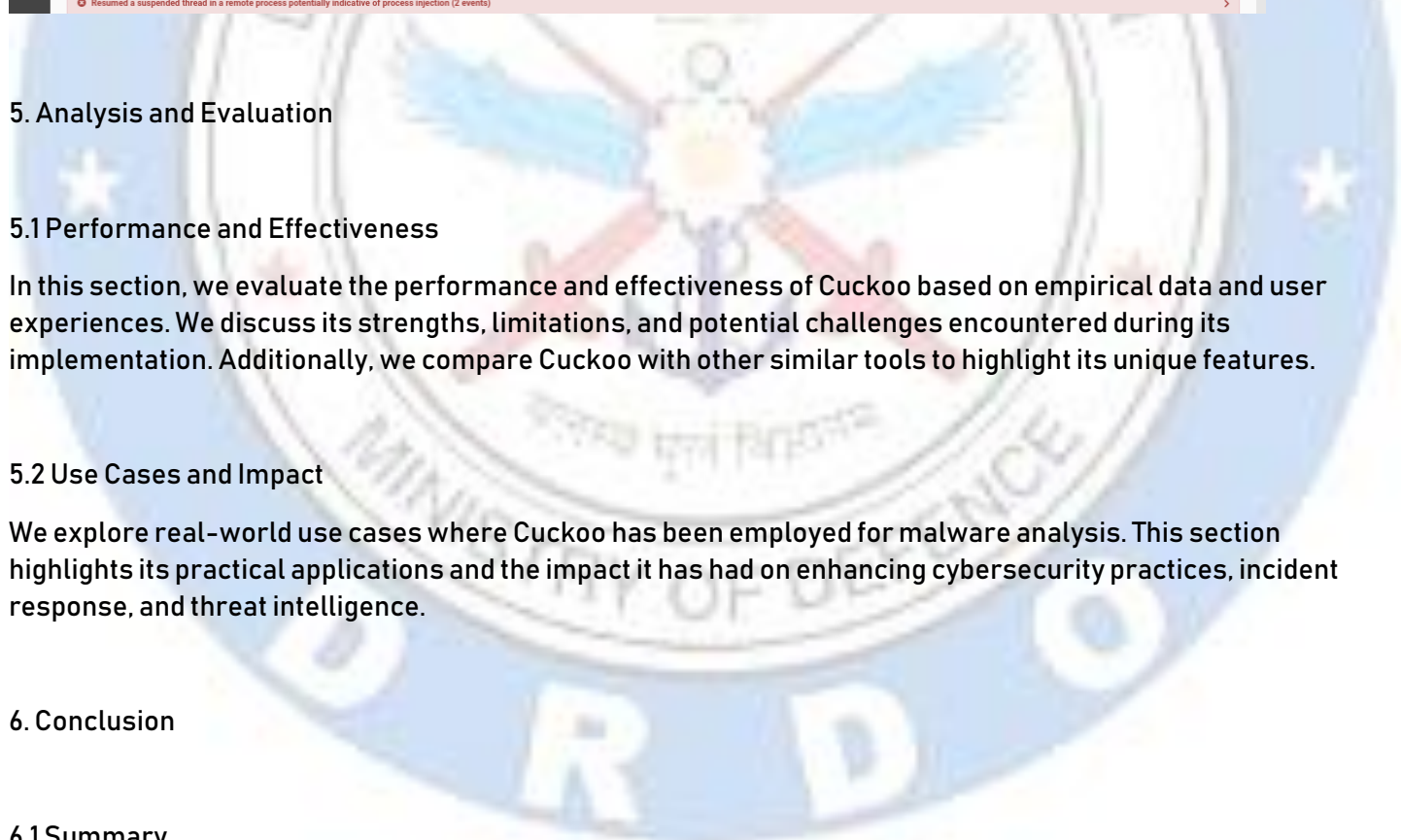
cuckoo web --host 127.0.0.1 --port 8080

vmcloak-vboxnet0

sudo sysctl -w net.ipv4.conf.vboxnet0.forwarding=1

sudo sysctl -w net.ipv4.conf.wlp13s0.forwarding=1

cuckoo --debug
```

5. Analysis and Evaluation

5.1 Performance and Effectiveness

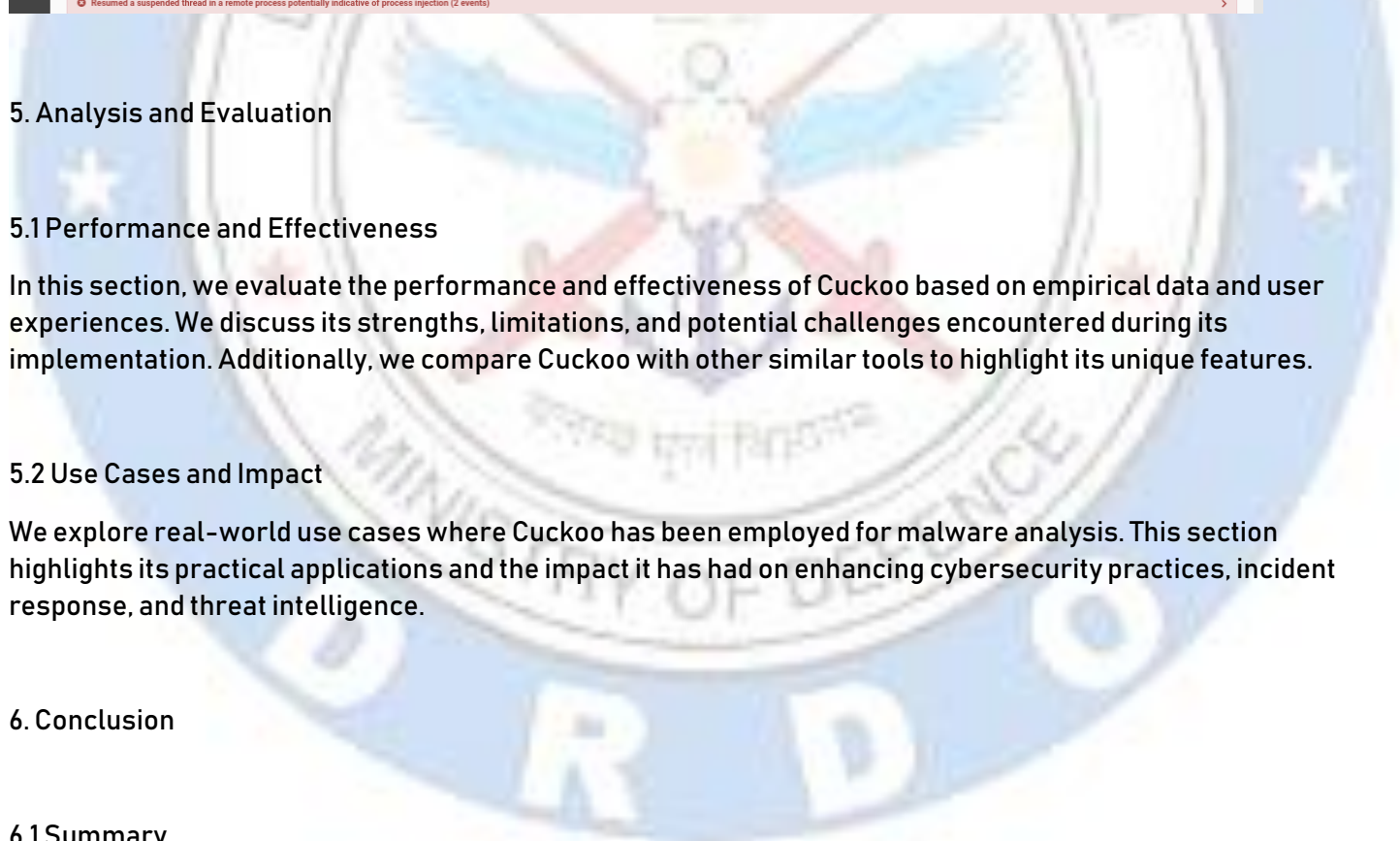
In this section, we evaluate the performance and effectiveness of Cuckoo based on empirical data and user experiences. We discuss its strengths, limitations, and potential challenges encountered during its implementation. Additionally, we compare Cuckoo with other similar tools to highlight its unique features.

5.2 Use Cases and Impact

We explore real-world use cases where Cuckoo has been employed for malware analysis. This section highlights its practical applications and the impact it has had on enhancing cybersecurity practices, incident response, and threat intelligence.

6. Conclusion

6.1 Summary



Resumed a suspended thread in a remote process potentially indicative of process injection (2 events)

5. Analysis and Evaluation

5. Analysis and Evaluation

5.1 Performance and Effectiveness

In this section, we evaluate the performance and effectiveness of Cuckoo based on empirical data and user experiences. We discuss its strengths, limitations, and potential challenges encountered during its implementation. Additionally, we compare Cuckoo with other similar tools to highlight its unique features.

5. Analysis and Evaluation

5.1 Performance and Effectiveness

In this section, we evaluate the performance and effectiveness of Cuckoo based on empirical data and user experiences. We discuss its strengths, limitations, and potential challenges encountered during its implementation. Additionally, we compare Cuckoo with other similar tools to highlight its unique features.

5. Analysis and Evaluation

5.1 Performance and Effectiveness

In this section, we evaluate the performance and effectiveness of Cuckoo based on empirical data and user experiences. We discuss its strengths, limitations, and potential challenges encountered during its implementation. Additionally, we compare Cuckoo with other similar tools to highlight its unique features.

5.2 Use Cases and Impact

We explore real-world use cases where Cuckoo has been employed for malware analysis. This section highlights its practical applications and the impact it has had on enhancing cybersecurity practices, incident response, and threat intelligence.

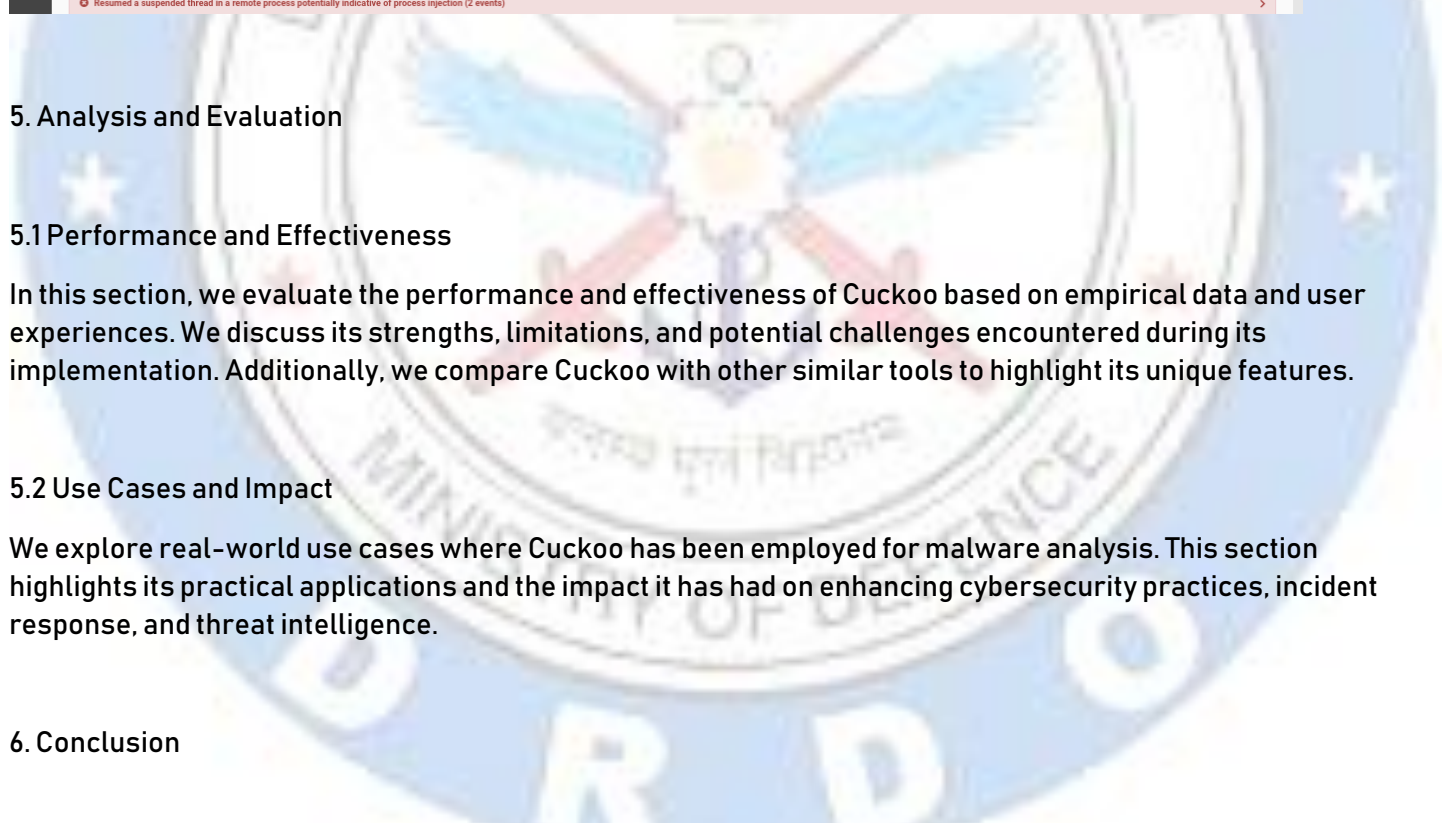
5. Analysis and Evaluation

5.1 Performance and Effectiveness

In this section, we evaluate the performance and effectiveness of Cuckoo based on empirical data and user experiences. We discuss its strengths, limitations, and potential challenges encountered during its implementation. Additionally, we compare Cuckoo with other similar tools to highlight its unique features.

5.2 Use Cases and Impact

We explore real-world use cases where Cuckoo has been employed for malware analysis. This section highlights its practical applications and the impact it has had on enhancing cybersecurity practices, incident response, and threat intelligence.



5. Analysis and Evaluation

5.1 Performance and Effectiveness

In this section, we evaluate the performance and effectiveness of Cuckoo based on empirical data and user experiences. We discuss its strengths, limitations, and potential challenges encountered during its implementation. Additionally, we compare Cuckoo with other similar tools to highlight its unique features.

5.2 Use Cases and Impact

We explore real-world use cases where Cuckoo has been employed for malware analysis. This section highlights its practical applications and the impact it has had on enhancing cybersecurity practices, incident response, and threat intelligence.

6. Conclusion



5. Analysis and Evaluation

5.1 Performance and Effectiveness

In this section, we evaluate the performance and effectiveness of Cuckoo based on empirical data and user experiences. We discuss its strengths, limitations, and potential challenges encountered during its implementation. Additionally, we compare Cuckoo with other similar tools to highlight its unique features.

5.2 Use Cases and Impact

We explore real-world use cases where Cuckoo has been employed for malware analysis. This section highlights its practical applications and the impact it has had on enhancing cybersecurity practices, incident response, and threat intelligence.

6. Conclusion

6.1 Summary

This section summarizes the research findings, emphasizing the working parameters and functionality of Cuckoo as an open-source malware analysis tool. It reinforces the significance of Cuckoo in addressing the challenges posed by malware and its contribution to the field of cybersecurity.

6.2 Future Directions

Based on the insights gained from this research, we identify potential areas for future enhancements and research. This includes the exploration of advanced analysis techniques, integration with additional security tools, and the continuous development and improvement of Cuckoo's capabilities

By examining the working parameters and functionality of Cuckoo, this research report aims to contribute to the broader understanding of open-source malware analysis tools and their implications in the domain of cybersecurity.

