|  |  |
| --- | --- |
|  | **MINISTRY OF EDUCATION AND TRAINING** |

|  |
| --- |
| **FPT UNIVERSITY** |
| CapStone Project Document |
| [Virus Scanning Application] |
|  |
| |  |  |  | | --- | --- | --- | | **ViPig** | | | | **Group Member** | Trần Trung Duy  Lê Nguyên  Nguyễn Công Hà Sa  Trần Văn Toản  Vũ Hồng Hà | SE00000  SE03803  SE03384  SE03872  SE03912 | | **Supervisor** | Phạm Minh Thuấn | | |
|  |

|  |
| --- |
| - Hanoi, 08/16 - **Report 1****INTRODUCTION****Purpose of this document**In this document, we are going to describe about the developer group, the initial idea of ours Project, posing an actual cyber security problem, and overview of this Project, it’s objectives as well as benefit for both our group and the customers of this project.**About Us**We are five student of FPT University that want to graduate as well as giving people more options to choose in cyber environment.**Overview****Malware Overview** Malware, short for malicious software, is an umbrella term used to refer to a variety of forms of hostile or intrusive software, including [computer viruses](https://en.wikipedia.org/wiki/Computer_virus), [worms](https://en.wikipedia.org/wiki/Computer_worm), [trojan horses](https://en.wikipedia.org/wiki/Trojan_horse_(computing)), [ransomware](https://en.wikipedia.org/wiki/Ransomware_(malware)), [spyware](https://en.wikipedia.org/wiki/Spyware), [adware](https://en.wikipedia.org/wiki/Adware), [scareware](https://en.wikipedia.org/wiki/Scareware), and other malicious programs. It can take the form of [executable code](https://en.wikipedia.org/wiki/Executable_code), [scripts](https://en.wikipedia.org/wiki/Script_(computing)), active content, and other software.  In the past 2 years ( 2016-2017) Cyber attackers reach a new level of ambition, massive attacks, including multi-million dollar virtual bank heists, overt attempts to disrupt the US electoral process by state-sponsored groups, and some of the biggest distributed denial of service (DDoS) attacks on record powered by a botnet of Internet of Things (IoT) devices, 2017 started with the rise of shadow brokers – a mysterious hacking group claimed to have breached the spy tools of the elite NSA-linked operation and release of particularly significant alleged NSA tools, including a Windows exploit known as EternalBlue, which facilitate others hackers to perform 2 global ransomware attacks: Wannacry and Petya.  Cyber attacks involving sabotage have traditionally been quite rare, but 2016 saw two separate waves of attacks involving destructive malware. Disk-wiping malware was used against targets in Ukraine in January and again in December, attacks which also resulted in power outages. Meanwhile the disk-wiping Trojan Shamoon reappeared after a four-year absence and was used against multiple organizations in Saudi Arabia. Only a year later, another two waves of attack, which also involving malware, or rather, ransomware hit the cyber world. Although new kind of attacks has been developed, which used very simple tools and tactics to make a big impact. Attackers rely on straightforward approaches, such as spear-phishing emails and “living off the land” by using whatever tools are on hand, such as legitimate network administration software and operating system features, malicious sofeware is still growing.  Some data through the years  **Cuckoo**Cuckoo, or Cuckoo Sandbox, is an open source automated malware analysis system for automating analysis of suspicious files. To do so it makes use of custom components that monitor the behavior of the malicious processes while running in an isolated environment.It can retrieve the following type of results:  * Traces of calls performed by all processes spawned by the malware. * Files being created, deleted and downloaded by the malware during its execution. * Memory dumps of the malware processes. * Network traffic trace in PCAP format. * Screenshots taken during the execution of the malware. * Full memory dumps of the machines. * Cuckoo is designed to be used both as a standalone application as well as to be integrated in larger frameworks, thanks to its extremely modular design.   It can be used to analyze:   * Generic Windows executables * 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 * Almost anything else  **Architecture** Cuckoo Sandbox consists of a central management software which handles sample execution and analysis.  Each analysis is launched in a fresh and isolated virtual or physical machine. 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.  The following picture explains Cuckoo’s main architecture:  ../../_images/architecture-main.png   1. **CUSTOMIZATION**  Auxiliary Modules **Auxiliary** modules define some procedures that need to be executed in parallel to every single analysis process. All auxiliary modules should be placed under the *cuckoo/cuckoo/auxiliary/* directory, that way the module will fall under the *cuckoo.auxiliary* module. Machinery Modules **Machinery** modules define how Cuckoo should interact with your virtualization software (or potentially even with physical disk imaging solutions). Since we decided to not enforce any particular vendor, from release 0.4 you are able to use your preferred solution and, in case it’s not supported by default, write a custom Python module that defines how to make Cuckoo use it.  Every machinery module should be located inside the *cuckoo/cuckoo/machinery/* directory so that it will fall under the *cuckoo.machinery* module.  The only requirements for Cuckoo are that:   * The class inherits from *M achinery*. * You have a *start()* and *stop()* functions. * You raise *CuckooMachineError* when something fails.   As you understand, the machinery module is a core part of a Cuckoo setup, therefore make sure to spend enough time debugging your code and make it solid and resistant to any unexpected error. Analysis Packages As explained in [Analysis Packages](http://docs.cuckoosandbox.org/en/latest/usage/packages/), analysis packages are structured Python classes that describe how Cuckoo’s analyzer component should conduct the analysis procedure for a given file inside the guest environment.  As you already know, you can create your own packages and add them along with the default ones. Designing new packages is very easy and requires just a minimal understanding of programming and of the Python language. Processing Modules Cuckoo’s processing modules are Python scripts that let you define custom ways to analyze the raw results generated by the sandbox and append some information to a global container that will be later used by the signatures and the reporting modules.  You can create as many modules as you want, as long as they follow a predefined structure that we will present in this chapter. Signatures With Cuckoo you’re able to create some customized signatures that you can run against the analysis results in order to identify some predefined pattern that might represent a particular malicious behavior or an indicator you’re interested in.  These signatures are very useful to give a context to the analyses: both because they simplify the interpretation of the results as well as for automatically identifying malware samples of interest.  Some examples of what you can use Cuckoo’s signatures for:   * Identify a particular malware family you’re interested in by isolating some unique behaviors (like file names or mutexes). * Spot interesting modifications the malware performs on the system, such as installation of device drivers. * Identify particular malware categories, such as Banking Trojans or Ransomware by isolating typical actions commonly performed by those. * Classify samples into the categories malware/unknown (it is not possible to identify clean samples)   You can find signatures created by us and by other Cuckoo users on our [Community](https://github.com/cuckoosandbox/community) repository. Reporting Modules After the raw analysis results have been processed and abstracted by the processing modules and the global container is generated (ref. [Processing Modules](http://docs.cuckoosandbox.org/en/latest/customization/processing/)), it is passed over by Cuckoo to all the reporting modules available, which will make use of it and will make it accessible and consumable in different formats. **Meteor** Meteor is a full-stack JavaScript platform for developing modern web and mobile applications. Meteor includes a key set of technologies for building connected-client reactive applications, a build tool, and a curated set of packages from the Node.js and general JavaScript community.  C:\Users\Sa Nguyen\Desktop\meteor.png   * Fundamentals of Meteor: * **Data on the Wire**. Meteor doesn’t send HTML over the network. The server sends data and lets the client render it. * **One Language**. Meteor lets you write both the client and the server parts of your application in JavaScript. * **Database Everywhere**. You can use the same methods to access your database from the client or the server. * **Latency Compensation**. On the client, Meteor prefetches data and simulates models to make it look like server method calls return instantly. * **Full Stack Reactivity**. In Meteor, real-time is the default. All layers, from database to template, update themselves automatically when necessary. * **Embrace the Ecosystem**. Meteor is open source and integrates with existing open source tools and frameworks. * **Simplicity Equals Productivity**. The best way to make something seem simple is to have it actually be simple. Meteor’s main functionality has clean, classically beautiful APIs. * List of features: * Meteor not only has a one-step installation for configuration and setup, but it also has an isomorphic API, which refers to using the same code on the frontend or backend, or even for mobile and web apps. This saves developers hours, perhaps even days and weeks, since there is no need for developers to wrestle with installing, configuring, and learning disparate libraries, module managers, multifarious APIs, drivers, and the like. * It offers not only a front-end framework, like Backbone.js, but also a backend that seamlessly integrates with the frontend, and an easy-to-use API for communicating between the two; this provides developers with straightforward, no-fuss client-server data management (Collections, Models, etc.), server-side methods, and server session management. * It provides not only bidirectional persistent communication (like socket.io), but also simplified reactive programming (like Bacon.js). The reactive programming library works in conjunction with the front-end framework to reactively (that is, instantly and continuously) update the UI whenever dependable data or variables change. Moreover, a Meteor community developer has implemented the Meteor front-end templating engine on the server, providing server side templating for Meteor. * It offers not only a stack that includes MongoDB database (with PostgreSQL and others planned), but also a front-end representation of MongoDB, called Minimongo, written entirely in JavaScript and available in every connected client. Meteor integrates the two (MongoDB on the backend and Minimongo on the frontend) in a well-conceived manner to mitigate latency, a concept called latency compensation. This results in considerably faster page updates and reloads, leading to a more satisfying user experience for developers and end users alike. * It not only has a standardized Mongo API on both the frontend and backend, but it also has Oplog tailing for MongoDB, resulting in applications using substantially less server resources. * It has a standard front-end router (created by a Meteor community member) that implements the best features from other popular front-end routers, and this router also provides server side routing and includes a familiar API like Express.js’s routing API, even allowing for connect middleware, RESTful endpoints, and the like. * It has a lightweight front-end framework, Blaze.js that has a templating engine that supersedes Handlebars.js, with a rendering engine similar to the WIP (Work in Project) HTMLBars. I should note that Blaze is neither as feature-rich and robust as Ember.js nor as expansive and extensible as Angular.js. * Its integrated live browser reload (also known as hot code load and hot code push) not only automatically reloads your live web page whenever you make development changes on the frontend (HTML, CSS, images, JavaScript, etc.), but it also automatically refreshes just the necessary DOM elements on the page (without reloading the entire page), even when there are dependent changes to data on the backend (MongoDB) or frontend (Minimongo). * It comes with requisite core packages to handle minification, preprocessing, concatenation, OAuth and custom user authentication (signup, login, forget password, etc.), emailing, and coffeescripting; packages for popular frameworks like bootstrap, backbone, and jQuery; and even a package for SEO compatibility. * It has its own command line tool that provides many of the functionalities provided by tools such as grunt, NPM, NVM, and the like. And if you add EventedMind’s em5scaffolding tool, you also have some (though not most) of the functionalities provided by popular scaffolding tools prominent in Rails and Yeoman. * It allows you to use NPM modules and it provides its own build system (a custom package manger) that transcends NPM, providing nearly all the worthwhile and crucial NPM functionalities and more. You can install third-party or custom Meteor packages from atmospherejs.com, the official repository for Meteor packages. * It uses synchronous style coding, courtesy of the Fibers JavaScript library. This provides an easy to read code structure that many find more appealing than the asynchronous structure of callback functions, common in most JavaScript frameworks. I should caution that you must pay close attention to the section in the road map that deals with making async calls in Meteor because many seem to have trouble when making async calls in Meteor. * It even provides support for mobile apps through Cordova Phonegap integration. You can easily deploy your Meteor app as a mobile app, using these simple commands:     *Runs your Meteor app in an iOS simulator and starts the server*  Or    *Runs your Meteor app in an Android simulator and starts the server*   * Meteor takes a proactive approach to security, reportedly with a core developer dedicated to addressing security. * It has an official testing framework, Velocity. With Velocity, you can use your favorite testing frameworks like Jasmine or Mocha, and run acceptance tests with Selenium.  **React** React was created by Jordan Walke, a software engineer at Facebook. He was influenced by XHP, an HTML component framework for PHP. It was first deployed on Facebook's newsfeed in 2011 and later on Instagram.com in 2012. It was open-sourced at JSConf US in May 2013. React Native, which enables native iOS, Android and UWP development with React, was announced at Facebook's React.js Conf in February 2015 and open-sourced in March 2015. On April 18, 2017, Facebook announced React Fiber, a new core algorithm of React framework library for building user interfaces. React Fiber will become the foundation of any future improvements and feature development of the React framework.    Image result for react js   * React (sometimes styled React.js or ReactJS) is an open-source JavaScript library for building user interfaces. * It is maintained by Facebook, Instagram and a community of individual developers and corporations. According to JavaScript analytics service Libscore, React is currently being used on the websites of Netflix, Imgur, Buffer, Bleacher Report, Feedly, Airbnb, SeatGeek, HelloSign, Walmart, and others. * React allows developers to create large web applications that use data which can change over time, without reloading the page. Its main goal is to be fast, simple and scalable. React processes only user interfaces in applications. This corresponds to View in the Model-View-Controller (MVC) template, and can be used in combination with other JavaScript libraries or frameworks in MVC, such as AngularJS. * Features of React: * **One-way data flow**. Properties, a set of immutable values, are passed to a component's renderer as properties in its HTML tag. A component cannot directly modify any properties passed to it, but can be passed callback functions that do modify values. This mechanism's promise is expressed as "properties flow down; actions flow up". * **Virtual DOM**. Another notable feature is the use of a "virtual Document Object Model", or "virtual DOM". React creates an in-memory data structure cache, computes the resulting differences, and then updates the browser's displayed DOM efficiently. This allows the programmer to write code as if the entire page is rendered on each change, while the React libraries only render subcomponents that actually change. * **JSX**. React components are typically written in JSX, a JavaScript extension syntax allowing quoting of HTML and using HTML tag syntax to render subcomponents. This is a React-specific grammar extension to JavaScript like the now-defunct E4X. HTML syntax is processed into JavaScript calls of the React framework. Developers may also write in pure JavaScript. JSX is similar to another extension syntax created by Facebook for PHP, XHP. JSX looks like regular HTML. * **Architecture beyond HTML**. The basic architecture of React applies beyond rendering HTML in the browser. For example, Facebook has dynamic charts that render to *<canvas>* tags, and Netflix and PayPal use isomorphic loading to render identical HTML on both the server and client. * **React Native**. React Native libraries were announced by Facebook in 2015, providing the React architecture to native iOS, Android and UWP applications.  **Other tools****Project review****Overview of ideas** The whole idea is to develop an automated analysis malware-analysis system, which we currently use open-source solution – Cuckoo Sandbox. We can separate Cuckoo Sandbox into 3 main repositories so that we can continually inherit and develop the project, which are:   * Cuckoo, the main repository * Monitor, to develop the hooking and injection technique on analysis machines * Community, customized signatures for malwares and integrated plugins for Snort, Suricata, etc.   With Cuckoo, we could solve those problems:   * Analysis malwares using their behaviors on multiple isolated environments, ex: Windows, Linux, Android, etc. We currently used only Windows as analysis machine. * Traces of calls performed by all processes spawned by the malware. * Files being created, deleted and downloaded by the malware during its execution. * Memory dumps of the malware processes. * Network traffic trace in PCAP format. * Screenshots taken during the execution of the malware. * Full memory dumps of the machines.   Application base on Cuckoo structure, where file will be examine in a safe environment call analysis machine, analysis result will be sent to result server where it is analyzed again with database from an api server, the final result will be show in the web interface.  D:\vsc 8.1\Desktop\1.png   * 1. **Objectives**   Successfully install and configure Cuckoo as well as develop new modules:   * Videos recording module * Signature module   Research, understand how Cuckoo work: how it process, available, modules, develop it’s system architecture.  Develop a new web interface to connect with Cuckoo **BENEFIT FROM OUR PROJECT****For group**  * Learning about cuckoo sandbox * Experiences in project management: manage plan, times management, members management, risks assessment, working in pressure environment. * Know how to work virtualization service such as google cloud ( Firewall, Compute engine, routing) * Working with Ubuntu, CenOS, VPN * Increase group working skill  **For customers** Another online malware scanning tools with new features ( Vietnamese interface) |
| **III. Report No.3: Risk Assessment**   1. **The Need of Assessment**   Every project contains project risk hidden inside it. With software development project, it depends heavily on the amount of risk that corresponds to each project activity. It’s not enough to be aware of the risks. To achieve a successful outcome, project leadership must identify, assess, prioritize, and manage all of the major risks. Performing a risk assessment is an important step in being prepared for potential problems that can occur within any software project. During the risk assessment, if a potential risk is identified, a solution or plan of action should be developed. Effective analysis of risks will help to effective planning and assignments of work.   1. **Identify Critical Information Assets**    1. Information Asset Classification       1. The Probability of a threat exploiting a vulnerability in an asset   Our website was built on a Cuckoo platform on Linux Operation System. So, it exists vulnerabilities can be exploited.   * On server: malware, unpatched software, information gathering, DDoS, brute-force, social engineering for careless user. * On Website: XSS vulnerability.   + 1. The Impact of a threat exploiting a vulnerability in an asset usually measured in terms of cost to the asset’s stakeholders   When the system is compromised, it can cause damage to system only as well as information of users.   * System: * Reduce performance. * Loss of credibility. * Cost for fix vulnerabilities. * Users: * Extorted to redeem leaked information.   1. System Characterization      1. Logical Architecture: security domains, how data is stored, transmitted and processed         + Image goes here         + Phase 1: User submits files or URLs to web server         + Phase 2: Web server call API of task distribution server to submit tasks         + Phase 3: Task distribution server sent tasks to analysis machine ( we only Windows for analysis machine)         + Phase 4: Analysis machine return information about analyzed tasks to task distribution server, then distribution tasks server will save to database         + Phase 5: Web server return information of analyzed tasks in database to client      2. System Components: hardware, software, network environment, servers, switches, firewalls, OS, applications, databases  |  |  |  |  | | --- | --- | --- | --- | | Functional server | Item | Description | Note | | Web server + Task distribution server + Database server + Analysis server(s) | Virtualization software | VSphere | On physical machine | | Web server | OS | Ubuntu 16.04 |  | |  | Application | Meteor, nodejs |  | | Task distribution server | OS | Ubuntu 16.04 |  | |  | Application | Cuckoo Distributed, tcpdump, mitmproxy |  | | Analysis server | OS | Windows 7 SP1 64bit, Windows XP 32bit |  | |  | Application |  | Depend on the need of malware types | | Database server | Database | Mongodb |  |  * + 1. Users of the System     2. Security and Compliance Requirements: *Confidentiality, Integrity, Authentication, Non-repudiation, Availability*, relevant laws, regulations…     3. Information Protection Priorities  1. **Risk Identification**    1. Threat Identification       * + Information disclosure   Information disclosure allows the hacker to access sensitive information of the systems. Therefore, user always consider what their information is going to public and whether it can be collected by a hacker. For example, when a hacker gets information about a server's OS. Old operating systems usually contain errors that had been announced which is huge advantage for hacker to attack system.   * + - * Misconfiguration   It is called misconfiguration when a component in a system is compromised due to an unsafe configuration. A web application will have a lot of different layers when being built so the Misconfiguration occurs very common and at various levels. For example, when the system admin configures the network and accidentally leaves some ports open to access to open-web applications, the attackers could exploit the vulnerability to take control of the system or to perform a bigger attack.   * + - * Bypass authentication   Most web applications nowadays have a user authentication mechanism to ensure system's security. That's why authenticity has become the primary attack target of attackers. This vulnerability allows attackers to infiltrate the system, bypass the authentication mechanism to perform privilege escalation.   * 1. Vulnerability Identification      + - **Currently we have 3 main servers: web, task distribution, database server.**        - **Web server:**        - **Client side: React JS (XSS)**        - **DDOS**        - **Task distribution server”**        - **Privilege escalation (v** [**https://www.exploit-db.com/exploits/39772**](https://www.exploit-db.com/exploits/39772)**)**        - **Bruteforce authentication port (SSH)**        - **DOS**        - **Database server:**        - **Bruteforce authentication**        - **DOS**  1. **Risk Analysis**    1. Impact Assessment    2. Likelihood Assessment    3. Risk Determination (Rating)       1. Risk-Level Matrix       2. Description of Risk Level 2. **Control Identification and Assessment**    1. Control Methods       1. Technical (safeguards, tools)       2. Non-technical (management and operational controls)    2. Control Types (Deterrent, Preventive, Detective, Corrective…)    3. Residual Risk Evaluation    4. Risk Monitoring and Controlling   **IV. Report No.4: RMP (Risk Assessment Plan)**   1. **Objectives**    1. Lists of Threats/Vulnerabilities    2. Costs associated with risks    3. List of Recommendations to Reduce the Risks    4. Costs Associated with Recommendations    5. CBA (Cost-Benefit Analysis) 2. **Assigning Responsibilities** 3. **Describing Procedures and Schedules for Accomplishment** 4. **Reporting Requirements**    1. Present Recommendations    2. Document Management Response to Recommendations    3. Document and Track Implementation of Accepted Recommendations 5. **Plan of Action and Milestones** 6. **Charting the Progress of a RMP**    1. Milestone Plan Chart    2. Gantt Chart    3. Critical Path 7. **Tools and Practices**    1. Framework (Risk IT, GAISP, CobiT, PCI DSS, ISO 17799, ISO/IEC 27002, NIST Handbook, etc.)    2. Policies, Procedures, Plans, and Processes, Including CBA   **V. Report No.5: DIP**   1. **Risk Response Planning**    1. Major Risk Treatment: after monitoring divide major risks into avoiding, mitigating, accepting, transferring (outsourcing, etc.)    2. Risk Mitigation Treatment (consider prevention, detection, and response)    3. Risk Mitigation Plan (RMiP)       1. Cost and Time to Implement       2. Operational Impact 2. **Priority Risk Mitigation List**    1. Threat/Vulnerability Matrix Method    2. Prioritizing Countermeasures    3. Verify How They Can Be Mitigated 3. **Perform CBA on the Identified List**    1. Calculate CBA    2. CBA Report 4. **Implement the RMiP**    1. Tools and Techniques (Algorithms, Firewalls, InfoSec Softwares, etc.)    2. Policies, Procedures for Controlling Regular Backups and Configuration Hardening    3. Operational Controls (Employee Training in Security Awareness, Configuration Management, Contingency Planning, Incident Response, etc.) 5. **Follow Up on the RMiP**    1. Ensuring Countermeasures Are Implemented    2. Ensuring Security Gaps Have Been Closed   **VI. Report No.6: VD**   1. **Repeat Risk Assessment Process**    1. Check and Add for a New Critical Asset Appeared    2. Check for a Change of IT Environment    3. New Risk Assessment 2. **Risk Analysis**    1. Qualitative Analysis    2. Quantitative Analysis    3. Provable Risk Mitigation (provided no New Major Risks appeared)   VII. Reversing techniques  PE FILE  ● PE (Portable Executable) is the file format for Windows' executable binaries – You can find imported libraries/functions from the PE headers.  ● 3 conventional ways to use libraries :  – Dynamic link at compile time: .dll files are loaded into the memory space of a process at load time, and the main executable just calls the needed functions in the DLLs  – LoadLibrary at run time: .dll files are loaded into the memory space of a process on run time  – Static link at compile time: .lib files are combined into a PE file to make a big fat file that doesn't have external dependencies.    **PACKERS**  ● Originally used to compress executables back when disk space was at a premium  ● The executable then decompresses itself in memory and runs as normal  ● Nowadays they are mostly used for obfuscating binaries. Specifically since all the data for the original binary is compressed and/or encrypted, it prevents analysts from being able to infer things about the binary based on strings or function imports  ● UPX, ASPack, MPRESS, Themida, etc.  ● For dynamic analysis, since we will actually execute a sample, this is not a hindrance      **WINDOWS LIBRARY FILE**   |  |  | | --- | --- | | DLL Name | Description | | Kernel32.dll | Provides APIs for memory management, file operations, process/thread creation | | User32.dll | Implements Windows USER component to provide graphical user interface such as menu bar, scroll bar, button, mouse pointer cursor, etc. | | GDI32.dll | Exports Graphics Device Interface functions for drawing, text output, font management, etc. | | Ntdll.dll | Interface to kernel for memory management, file operations, process/thread creation. It is not normally used by Windows applications directly | | Ws2\_32.dll | Exports Windows Sockets APIs | | Wininet.dll | Provides high level network API such as HttpOpenRequest and FtpGetFile |     IIX. Bugs   1. Grab screen fail    * + - <https://github.com/ViPig/cuckoo/issues/2> 2. Dump.pcap is always empty, probably tcpdump’s fault    * + - <https://github.com/ViPig/cuckoo/issues/3> 3. Command '['bin\\is32bit.exe', '-n', 'lsass.exe']' returned non-zero exit status 1    * + - <https://github.com/ViPig/cuckoo/issues/4> 4. some files on \appdata\local\temp permission denied    * + - <https://github.com/ViPig/cuckoo/issues/5> 5. Tasks added by Cuckoo Submit and API not updated in Cuckoo Web    * + - <https://github.com/ViPig/cuckoo/issues/11> 6. Pymongo connection reset by peer    * + - <https://github.com/ViPig/cuckoo/issues/12> 7. Threading problem with auxiliary module – screenrecord    * + - <https://github.com/ViPig/cuckoo/issues/13> 8. BsonParser lacking data    * + - <https://github.com/ViPig/cuckoo/issues/14> |
|  |