2022

C2 Microservices API: CH4RL3SCH4L3M4GN3

Thai H. Nguyễn
Sacred Heart University, tvhn.433@gmail.com

C^2 MICROSERVICES API: CH4RL3SCH4L3M4GN3

Thai H. Nguyen
Department of Cybersecurity
School of Computer Science & Engineering
Sacred Heart University
Fairfield, USA
nguyent62509@mail.sacredheart.edu

***Abstract*—In the 21st century, cyber-based attackers such as advance persistent threats are leveraging bots in the form of botnets to conduct a plethora of cyber-attacks. While there are several social engineering techniques used to get targets to unknowingly download these bots, it is the command-and-control techniques advance persistent threats use to control their bots that is of critical interest to the author. In this research paper, the author aims to develop a command-and-control microservice application programming interface infrastructure to facilitate botnet command-and-control attack simulations. To achieve this the author will develop a simple bot skeletal framework, utilize the latest in API development frameworks, and simulate 2 types of malicious cyber-attacks. The attacks will be in the form of data exfiltration and data encryption. The author realizes that there needs to be quantitative data aggregation on the performance of the API and malicious bots. The author will be designing and developing a system to achieve this goal as part of their future work.**

Keywords—Application Programming Interface, API, Advance Persistent Threat, APT, Bot, Command and Control, C², C&C, Cybersecurity, Data Encryption, Data Exfiltration, Malware, Social Engineering

I. INTRODUCTION

With the near seamless ubiquitous computing in current modern society, we interact with computing bots in greater ratios than with actual human beings. This is in the form of chatbots that perform mundane customer service or routine transactions. A recent Imperva, "Bad Bot Report 2021: The Pandemic of the Internet" report, states that they account for roughly 15.2% of Internet traffic. What's alarming with the report is the steadily increase in malicious bot traffic which accounted for 25.6% of Internet traffic, while humans took the rest with a not surprising 59.2%. As attackers become more sophisticated in their attack strategies, so too do the malicious bots they develop. Imperva categorizes these bots as Advanced Persistent Bots or APBs.

Based on research from the MITRE Foundation on the tactics, techniques, and procedures of Advance Persistent Threats use of malicious bots, their command-and-control mechanisms, and open-source intelligence, the author aims to design, architect, and engineer a command-and-control microservice application programming interface from which to simulate Advance Persistent Bot attacks.

The author chose this project direction to achieve the following:

1. Expand the author's skills, knowledge, and experience in designing, architecting, and engineering a full-scale application programming interface.
2. Strengthen my knowledge of Advanced Persistent Threat Tactics, Techniques, and Procedures.
3. Strengthen my knowledge of Command-and-Control Tactics, Techniques, and Procedures used by Advanced Persistent Threats.
4. Fortify the author's technical skillsets.
5. The author will be packaging this project into an API as a Service (APIaaS) and will only be proving this service on the Silk Road, a darknet marketplace. The author plans on only accepting all major cryptocurrencies.

The rest of the paper is organized as follows: Section 2 (Background & Related Work) provides an overview of the different technologies used in designing and developing the author's project, current research on Advanced Persistent Threats, bot and botnets, command-and-control, and social engineering. Section 3 (Proposed Work) provides an overview of the work the author is proposing, which include Application Programming Interface Infrastructure, Bot/Botnet Framework, Malware Capabilities, and Social Engineering Scenarios. Section 4 (Preliminary & Expected Results) provides an overview of the previous evaluations of current capabilities of the technologies being planned on utilizing in the project and the author's expected results for the remainder of the project. Finally, Section 5 (Conclusion & Future Work) concludes and details the author's future direction for the project.

II. BACKGROUND & RELATED WORK

In the following sub-sections of the Background & Related Work, the author provides an overview of the history and current status of application programming interface methodology, application programming interface frameworks such as Django and Flask, Advanced Persistent Threats, Bot & Botnets, Command-and-Control, and finally Social Engineering.

A. Application Programming Interface (API)

The modern usage of application programming interface can be defined as a connection between two or more applications, services, programs, systems, or other application programming interfaces, which are used to send and receive data and content. This is achieved through the simple POST and GET request functions of application programming interfaces. Application programming interface has been utilized since computing has been available to society. If it wasn't for Roy Fielding's doctoral dissertation call "Architectural Styles and the Design of Network-based Software Architectures" and technology companies such as Salesforce, eBay, and Amazon did application programming interfaces become widespread usage and evolution. Since its small beginnings in commercial applications, application programming interfaces now drive desktop applications, majority of web applications, mobile applications, and low-code to no-code applications. Below are two of the current application programming interface frameworks being utilized in modern application programming interface solutions.

1) Django

Django is a modern Python web application framework that supports a full-fledge application programming interface development environment. It was first developed between 2003 – 2005 by a team of web developers from a newspaper company. This framework emerged from the continued re-factoring and re-use of code they used to develop multiple websites for the company. They eventually packaged the framework into an open-source project in 2005. Django believes in enabling rapid development of secure and maintainable web applications. This is achieved through being fully packaged with all the tools and resources a developer needs to accomplish their development goals, being versatile by support multiple formats such as HTML, RSS, JSON, XML, YAML, and more, being secure by enabling defaults protections from popular web application attacks such as SQL injections, cross-site scripting, cross-site forgery, and clickjacking [9, 10].

2) Flask

Flask is another modern Python web application framework originally designed and developed by Armin Ronacher as an April Fool's Day joke in 2010, but quickly became a full fledge framework because of the popularity. It is considered more Pythonic and faster in development versus Django. This is due to its microframework designation, which means that the primary package is a barebones framework only including the essentials for web applications development and to gain features such as authentication, database ORM, input validation and sanitation, the developer must import, i.e., download those feature functionalities [11 – 16]. Flask is built on three primary technologies: Web Server Gateway Interface (WSGI), Werkzeug, and jinja2 [13 – 14]. A brief overview of these technologies follows:

a) Web Server Gateway Interface (WSGI): A Python standard for web application development, enabling common interface between web servers and web applications.

b) Werkzeug: A Web Server Gateway Interface (WSGI) toolkit that implements requests, response objects, and utility functality.

c) jinja2: A Python template engine used for web templating which combines a template with specified data sources to render dynamic webpages.

B. Advanced Persistent Threat (APT)

Advanced Persistent Threats are an evolution of the traditional practice of spying with enhanced capabilities due to technological advances. The first recorded research on Advanced Persistent Threat emerged back in 2010 outlined their characteristics including methods, detections, and defense techniques [1 – 8]. The findings from subsequent research in Advanced Persistent Threats detailed their general organization, sophistication, concealment, and purpose. While many Advanced Persistent Threats have different objectives and goals, they all share these characteristics [5 – 8]. Chen et al. [8] provides a detailed analysis of these characteristics in the following:

a) Organizing: purpose – APT actors are well-organized teams that appear to be directly or indirectly supported by governments, making them more resourceful and organized than traditional actors.

b) Sophistication: APTs are implemented through a series of properly plotted activities, from spear phishing, malware injection, data transmission to trace clearance. The multiplicity of attack channels and mechanisms significantly increased the success rate of compromise.

c) Concealment: actors lurk in the target network and disguise activities as normal ones and could update the malware without being noticed.

d) Purpose: traditional actors typically pursuit financial gains from victims, but APTs aim at confidential information in the firms or government agencies.

C. Bot/Botnet

As it relates to code-based software, a bot is a software that is programming to perform scheduled actions. The bot is primarily automated, which means once deployed it will do all the tasks without human interaction. There are several applications that a bot can perform, including chatbot-ing, web crawling, socialbot-ing, and being a malicious bot. The malicious bot is one of the author's primary focuses on this project [20 – 28]. Malicious bots can form several automated cyber-based actions (attacks) which is of great interest, they include:

a) Credential Stuffing

b) Web Scraping

c) Denial of Service (DoS) & Distributed Denial of Service (DDoS) Attacks

d) Brute Force Attacks

When multiple bots are linked together into a coordinated attacks such as Distributed Denial of Service Attacks via a command-and-control intermediate, they are considered botnets [20 – 28]. Based on research in botnet characteristics, lifecycle, and taxonomy [20 – 28], there are defining concepts which are critical to the author's development of

CH4RL3SCH4L3M4GN3. These concepts are the lifecycle of botnets [21 – 24] which are outlined below:

a) Spreading & Injection: This is the initial entry phase of the bot. The bot or attacker will utitlize different methods and techniques such as social engineering to infect the target with the bot.

b) Communications Stage (also considered the Command-and-Control Stage): In this stage the bot or botnet will establish communications to the command-and-control server. In other words, the bots or botnet will listen to the command-and-control servers or connect to them periodically to get new commands from the command-and-control server. A new command when detected by the bots or botnet is treated as an order which they will then execute the order and the results are reported to the command-and-control server; the bots or botnets then wait for new commands.

c) Attack Stage: In this final stage of the lifecycle the bot or botnet will execute the attack plan given by the command-and-control server. The bot or botnet will start to attack the targets according to the transmitted commands.

D. Command and Control (C^2)

In in the context of Cybersecurity, a command-and-control server is a computer (usually a server or multiple servers) under the control of malicious cyber-based attackers (usually Advanced Persistent Threats) [29 – 33]. Command-and-control servers are used primarily as the intermediary to communicate with malicious bots or botnets, disseminate malicious software, exfiltrate stole data, and conduct a multitude of other cyber-based attacks. There are a multiple of different command-and-control topologies used by malicious attackers [29 – 31]. These include the following:

a) Centralized (Hierarchical) Model: In a centralized or hierarchical model malicious attackers will user other bots or botnets as proxy servers for their command-and-control servers. It does reduce the number of bots or botnets that need to be aware of the location of the centralized server. It can be a compromised machine or a legitimate Internet service provider. When the victim is infected, it will communicate to the command-and-control server and then will wait or check for pending commands.

b) Decentralized (Random) Model: In a decentralized or random model the primary command-and-control server can send commands from any bot within the network. Attack commands are issued via an authoritative bot. These commands are often tagged as authoritative, which tells the issuing bot to automatically propagate the commands to all other bots or botnets.

c) Hybrid (Multi) Model: In a hybrid or multi model the malicious attacker can implement a combination of centralized and decentralized models. There are advantages and disadvantages of this implementation. The advantages are the attackers can hide their communications protocols and botnet architecture from detection. The disadvantages are the attackers must design, manage, and maintain new hybrid model designs as each successive design becomes detected and intelligence published on them.

There are several different communications protocols used by command-and-control servers to communicate with their bots or botnets []. These include the following:

d) Internet Relay Chat (IRC): Internet relay chat is mainly designed for group and simple client and server communication via communication mediums called channels. This is a very flexible protocol and with several open-source implementations of this protocol, enables malicious attackers to extend it in a way that suit their needs.

e) Hyper Text Transfer Protocol (HTTP) & Hyper Text Transfer Protocol Secure (HTTPS): HTTP & HTTPS is another communications protocol used by malicious attackers because enables malicious attackers to bypass security appliances. The main advantage of using the HTTP & HTTPS protocol is hiding bot and botnet traffics into normal endpoint user web traffics, so it can easily bypass firewalls with port-based filtering mechanisms and avoid IDS detection.

f) Peer-to-Peer (P2P): Recently more advanced bots and botnets have begun to use P2P decentralized communications. A variant of Phatbot used code from WASTE5 that implements an encrypted P2P protocol designed for private messaging and file transfer among a small number of trusted parties.

The author also surveyed a multitude of open-source command-and-control frameworks to evaluate their capabilities and effectiveness [31]. In the follow analysis, the author outlines each open-source command-and-control framework:

g) Metasploit: Metasploit is a very powerful tool which can be used by malicious attackers as well as ethical hackers to probe systematic vulnerabilities on networks and servers. Because it is an open-source framework, it can be easily customized and used with modern operating systems. Some of the capabilities available by Metasploit include command shell payloads that enable users to run scripts or random commands against a host, dynamic payloads that allow testers to generate unique payloads to evade antivirus software, and Static payloads that enable port forwarding and communications between networks.

h) Empire: Empire was created by Veris Group security practitioners Will Schroeder, Justin Warner, Matt Nelson and others in 2015. Empire is a unique attack framework in that its capabilities and behaviors closely resemble those used by current nation state advanced persistent threat actors. Empire is effective at evading security solutions, operating in a covert manner, and enabling attackers' total control over compromised systems. Empire command-and-control traffic is asynchronous, encrypted, and designed to blend in with normal network activity, making it exceptionally difficult for cybersecurity teams to identify Empire command-and-control traffic in the enterprise.

i) Pupy: Pupy is a cross-platform post-exploitation tool and capable of low detection operations. It's written in Python which makes it very convenient. Pupy can communicate using multiple kinds of transport, migrate into processes using reflective injection, and load remote python code, python packages and python C-extensions from memory.

E. Social Engineering (SE)

In the field of Cybersecurity, social engineering is the intersection of psychological manipulations through

technological and non-technological mediums. To provide a universal understanding on the definition of social engineering, the author will be referencing the extensive research from Muton et al. and their work on defining the social engineering domain [42]. Mouton et al. defines "Social Engineering" stating, "The science of using social interaction as a means to persuade an individual or an organization to comply with a specific request from an attacker where either the social interaction, the persuasion or the request involves a computer-related entity." Mouton et al. also defines the individual conducting the social engineering attack as well as the definition of the act of the attack. Mouton et al. defines a "Social engineer (noun)" as "An individual or group who performs an act of Social Engineering," and also a "Social engineer (verb)" as "To perform an act of Social Engineering. When the verb is used in the Past Perfect form, it means a successful Social Engineering attack has occurred. For example, "The target may not know that he or she has been social engineered."." While Mouton et al. defines a "Social Engineering attack" as "A Social Engineering attack employs either direct communication or indirect communication, and has a social engineer, a target, a medium, a goal, one or more principles and one or more techniques." The definition for a social engineering attack is also backed by Ivaturi et al.'s breaks-down of the taxonomy of social engineering attacks [41]. With these terms defined, readers will improve their understanding of the social engineering scenarios the author developed in Section 3.

In the next section, the author outlines their proposed work for the design, development, and engineering for the Application Programming Interface Infrastructure, Bot/Botnet Framework, Malware Capabilities, and Social Engineering Scenarios.

III. Proposed Work

A. Application Programming Interface Infrastructure

The author is proposing designing and developing CH4RL3SCH4L3M4GN3 application programming interface on the Flask and Flask-RESTful Framework. The author chose this Python Framework due to its scalability and lightweight packaging. This will facilitate the capabilities the author is aiming to develop for the Bot/Botnet Framework and Malware Capabilities with the timeframe of 5 – 6 months. These frameworks will allow the author to escalate design and develop, as well as test for edge cases and debug.

B. Bot/Botnet Framework

Based on prior and current research in bot and botnet architecture [20 – 28]. The author is proposing designing and developing a scaled-down architecture of current and next-generation bot and botnets. The author is aiming to have at least four bots communicating with the application programming interface. As seen in figure 1, CH4RL3SCH4L3M4GN3 is the primary application programming interface that will decentralize its capabilities to three servers, while also being able to communicate with each other. The initial communication server address will be pre-defined to the malicious bot but will be override once the malicious bot receives its attack instructions. The author is aiming to develop a communications obfuscation technique to increase the effectiveness of the malicious bot.

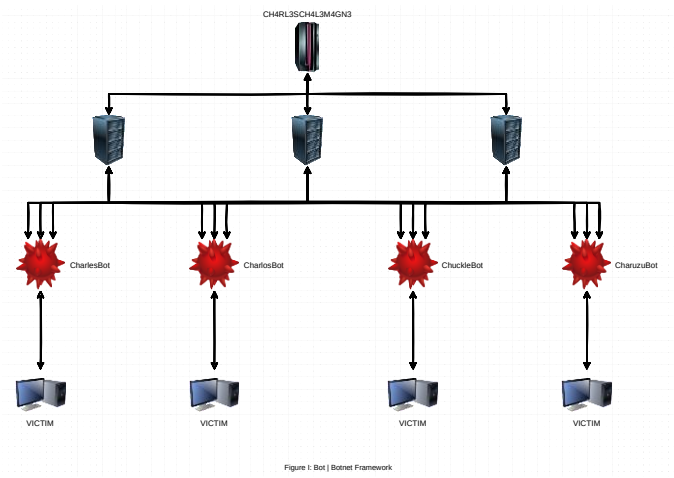

Figure I: Bot | Botnet Framework

C. Malware Capabilities

In the following two sections, Data Encryption and Data Exfiltration, the author is proposing designing and developing two malware capabilities that will be provided to the malicious bot by the application programming interface. The author chose data encryption and data exfiltration because research shows these are the primary attacks being carried out by Advanced Persistent Threats [1 – 8]. Data encryption is typically implemented in the form of Ransomware [1 – 8]. Data exfiltration takes the form of multiple mediums, examples of these include emails, http/https, and usb drives [37 – 40].

1) Data Encryption

The author is proposing designing a data encryption malware capability that will encrypt victim machines using symmetric encryption. The author chose this cryptographic algorithm because the author wishes to cause the most damage upon victim endpoints. The primary symmetric encryption the author will be utilizing will be AES [34 – 36]. To provide a tentative network communication of the data encryption attack bot, figure 2a provides this detail. In figure 2a, once the attack bot has achieved access to the victim machine, it will start a communications channel via HTTP/HTTPS to communicate to an identity repository to retrieve its attack identity. This is used to communicate back to the application programming interface; to retrieve its attack plan. The attack bot's identity is used to authenticate itself to the application programming interface. Once it has retrieved its data encryption attack instructions, it will commence its attack until complete. Once its attack has carried out, it will terminate communications to the application programming interface and kill itself.

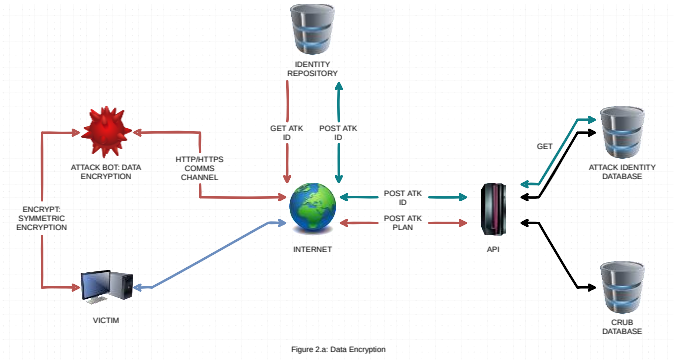

Figure 2.a: Data Encryption

To provide a more detailed understanding of the attack logic of the attack bot, figure 2b provides a flowchart of the data encryption instruction set. The attack bot will try to open a communications channel to the identity repository for a maximum of 5 tries, if it cannot communicate to the identity repository, it will terminate all processes and kill itself. If it can retrieve its identity from the repository, it will try to communicate to the application programming interface. It will try for a maximum of 5 tries, if it cannot communicate to the application programming interface, it will terminate all processes and kill itself. Finally, if it can retrieve its data encryption attack instructions, it will start encrypting its designated directories until complete. Once done it will terminate all processes and kill itself.

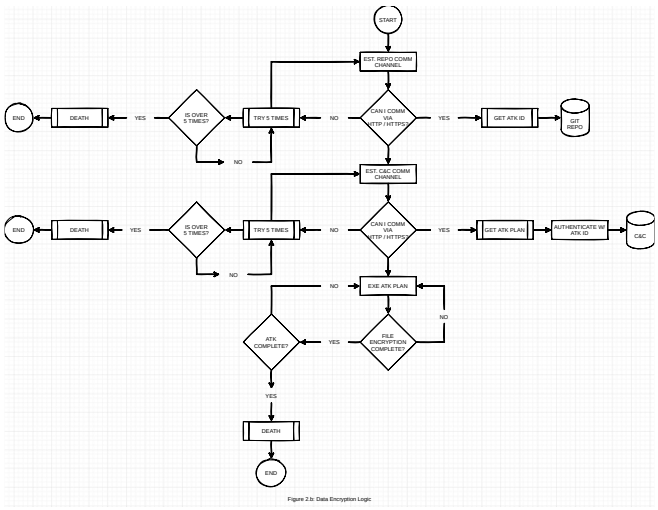

Figure 2.b: Data Encryption Logic

2) Data Exfiltration

The author is proposing designing a data exfiltration malware capability that will exfiltrate designated directories and files over HTTP/HTTPS [37 – 40]. Similar to the attack instructions from figure 2a, in figure 3a this attack includes an additional instruction set to post exfiltrated data back to the application programming interface. To give an overview again, once the attack bot has achieved access to the victim machine, it will start a communications channel via HTTP/HTTPS to communicate to an identity repository to retrieve its attack identity. This is used to communicate back to the application programming interface to retrieve its attack plan. The attack bot's identity is used to authenticate itself to the application programming interface. Once it has retrieved its data exfiltration attack instructions, it will commence its attack until complete. The malicious bot will exfiltrate 10 – 20 MB of data via Post Requests to the application programming interface. Once its target dataset has been exfiltrated, it will terminate all processes and communication channels to the application programming interface and finally kill itself.

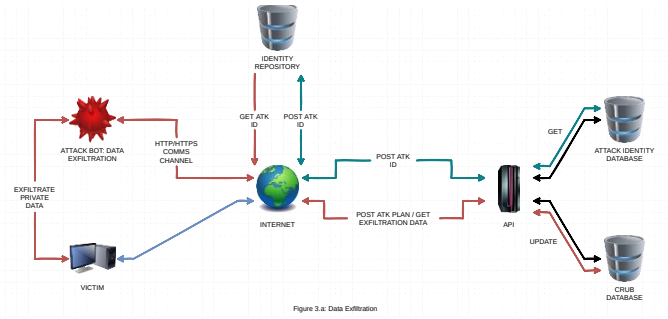

Figure 3.a: Data Exfiltration

Similar to the attack instructions from figure 2b, figure 3b provides a flowchart of the data exfiltration instruction set. The attack bot will try to open a communications channel to the identity repository for a maximum of 5 tries; if it cannot communicate to the identity repository, it will terminate all processes and kill itself. If it can retrieve its identity from the repository, it will try to communicate to the application programming interface. It will try for a maximum of 5 tries; if it cannot communicate to the application programming interface, it will terminate all processes and kill itself. Finally, if it can retrieve its data exfiltration attack instructions, it will start exfiltrating its designated directories until complete. Once done it will terminate all processes and kill itself.

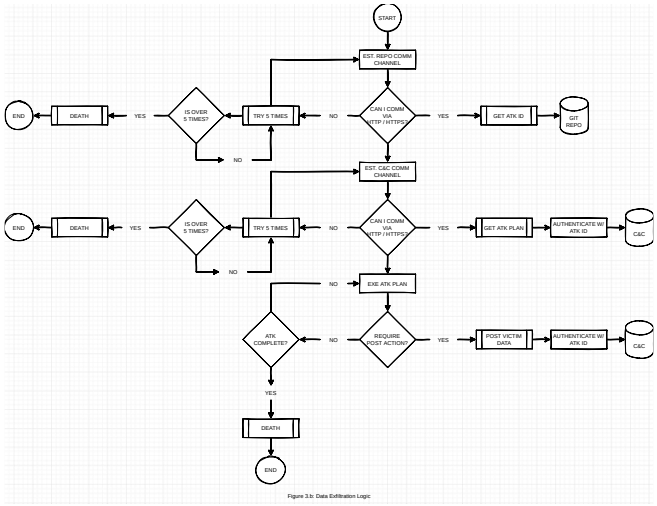

Figure 3.b: Data Exfiltration Logic

D. Social Engineering Scenario

Based on the social engineering attack scenario work outlined and proposed by Nguyen et al [42 – 45], the author has contributed to their social engineering attack scenario repository with two new social engineering attack scenarios. In the following the author created two specialized social engineering attack scenarios for data encryption and data exfiltration.

1) Higher Education Campus Vehicle Violation Ticketing Attack

Description: Social engineer creates replicated parking violation notices and places them onto student vehicles with the higher education institution's footprint. The owners of the vehicles find the notice and later navigate to the provided URL on the citation. The target is deceived to travers to a replicated malicious institutional portal, which will drive-by install a

malicious bot onto their computerized system. In addition, the malicious website will harvest students' institutional credentials.

Components

Communication Channel: Indirect Communication

Social Engineer: Social engineer is an individual

Target: Higher Education Institution Students

Medium: Parking Violation Citation, Technology-Based

Goal: Get victim to traverse to the malicious website, get malicious bot downloaded, and harvest credentials

Compliance Principles: Social Compliance & Authority

Technique: Phishing & Baiting

Phases

Step I: Attack Formulation

I. **Goal Identification:** The goal is to get victims to traverse to the malicious website, get malicious bot downloaded, and harvest credentials.

II. **Target Identification:** The target of the attack scenario are any students within the higher education institution that own vehicles.

Step II: Information Gathering

I. **Identify Potential Sources:** Potential intelligence sources include but not limited to higher education institution's public facing internet website, social media accounts, and physical reconnaissance. The institution's public facing website allows the SE to gather information on their specific parking violation policy, parking decal or makings, public institutional parking lot locations, student dormitory locations, faculty and employee designated parking lots, and institutional footprint maps. Institution's social media account will allow the SE to gather relevant information on the institution's public safety department and intelligence on their employees. Physical reconnaissance includes foot and vehicular reconnaissance. Reconnaissance allows the SE to monitor personnel movement, gathering locations, and visually inspect public safety personnel.

II. **Gather Information from Sources:** Gather intelligence from above mention potential sources that directly relate to the target's movement pattern, high traffic locations, areas of least security patrols, institutional parking decal or markings, and importantly obtain an authentic institutional parking violation ticket for replication.

III. **Assess Gathered Information:** Determine which location to scatter parking violation tickets that is relevant to the specific parking lot location, i.e., student lot, faculty lot, contractor lot, employee lot, and visitor lot. Ensure the simulated violation is within the standards and policies enforced within the institution's footprint. In parallel, determine if enough intelligence gathered to replicate the institution's public safety website, as well as list primary, secondary, and tertiary locations to maximize the attack zone.

Step III: Preparation

I. **Combination & Analysis of Gathered Information:** Determine the best time slots during which the SE can attempt to deploy the parking violation tickets in the SE's primary, secondary and tertiary locations. The SE will ensure to keep themselves anonymous whilst deploying the parking violation tickets, such as evading roaming security, close circuit cameras, and valid individuals of the institution that could question the SE's actions. It is also critical in deploying the parking violation tickets during a time of low human traffic in the attack zone.

II. **Development of an Attack Vector:** A Social engineer when developing the attack vector must detail the time and place of the attack to saturate the largest target population. The social engineer must ensure that the parking violation ticket is properly created, and the cloned institutional website must also be functioning and visually similar to the actual one. Social engineer ensures the malicious bot and credential harvesting capability is properly configured and functioning.

Step IV: Develop Relationship

I. **Establishment of Communication:** The act of deploying the parking violation tickets by the SE lasts until the target recognizes and obtains the parking violation ticket.

II. **Rapport Building:** The SE to properly build rapport, the SE must ensure the parking violation tickets looks similar to those that are typically used by the institutional organization. This includes but is not limited to the official branded logo on the ticket, working, contact information, and citations that are implicit within a higher education institutional organization.

Step V: Exploit Relationship

I. **Priming the Target:** The parking violation ticket should be realistic so that the owner of the vehicle will take it seriously and not simply dismiss it while driving home or to another location within the institution's footprint and mentally prepare to navigate to the institution's public safety website to either resolve or appeal the violation. The target is pressured to resolve the violation as soon as possible due to social compliance to do the right thing and the authority institutional public safety has over vehicle violations.

II. **Elicitation:** The "elicitation" step is almost implicit in this attack scenario. Most individuals will feel urged to comply with parking violation ticket requested actions. The provided institutional public safety website URL will allow the target to navigate to resolve the violation. Once the target

navigates to the malicious website, a malicious bot is automatically installed on the target's computerized system.

Step VI:

I. **Maintenance:** The parking violation ticket and institutional public safety website should be created in such a way that provides the target a sense of ease and trust. The parking violation ticket and website should replicate the real one utilized by the higher education institution. This will provide assurance to the target that the parking violation ticket is valid and their actions in resolving the violation is correct.

II. **Transition:** The SE utilizes the replicated institutional public safety website to automatically install a malicious bot to then pivot to other attack scenarios. Thus, the SE can then proceed to the "goal satisfaction" step.

III. **Goal Satisfaction:** The SE has attained their initial goal of gaining an unauthorized bot onto the target's computerized systems.

2) Higher Education Technology Lab Attack

Description: A social engineer pretends to be a student research assistant to a faculty member and convinces the institution's public safety officer to remotely unlock an unauthorized computer laboratory. SE gains access to the institution's computerized terminal and infects the entire laboratory computerized systems with a malicious bot.

Components

Communication Channel: Bidirectional Communication

Social Engineer: Social engineer is an individual

Target: Primary target: Institutional laboratory facility, Secondary target: Public Safety Department, Tertiary target: Institutional faculty member

Medium: Telephonic device, Technology-based

Goal: Gain unauthorized access to the institution's laboratory facility

Compliance Principles: Consistency Principle

Technique: Pretexting

Phases

Step I: Attack Formulation

I. **Goal Identification:** Primary goal is to gain unauthorized access to the institution's laboratory facility.

II. **Target Identification:** Primary target of the attack is the institution's laboratory facilities; secondary target is the institution's public safety officer, and the tertiary target is any faculty member that has the authority to access specified server room.

Step II: Information Gathering

I. **Identify Potential Sources:** Potential intelligence source is the institution's public website, social media accounts, public safety policies, information pertaining to the public safety department, physical reconnaissance. Physical reconnaissance includes drive-by and roaming reconnaissance of laboratory locations, and security implementations.

II. **Gather Information from Sources:** Gather all the information from the above sources. SE is required to find sources of information that directly relate to how to gain access to the unauthorized laboratory facility.

III. **Assess Gathered Information:** Determine the process in which to gain access to the unauthorized laboratory facility from the public safety department. Compile the specific information that faculty members are required to provide to the public safety officer. Type of faculty members that are authorized to access specified laboratory facility within the institution. Since it is technology-based medium, the SE will compile the types of information that is required by the public safety department to identify the faculty member. Detailed mapping and route of the location of all server facilities within the institution. Timeline of when faculty members are usually accessing the laboratory facility.

Step III: Preparation

I. **Combination & Analysis of Gathered Information:** Determine the specific laboratory facility, specific faculty member, time slot, and the required information to provide to public safety department.

II. **Development of an Attack Vector:** Develop an attack plan detailed to the specific time and location to conduct the attack. Prepare a detailed list of information required to provide the public safety department, and conversation script with primary, secondary and tertiary courses of action. Social engineer ensures the malicious bot is properly configured and functioning.

Step IV: Develop Relationship

I. **Establishment of Communication:** Social engineer initiates communication by physically calling the public safety office, up to the moment where the secondary target assists the attacker in unlocking the laboratory facility door.

II. **Rapport Building:** Social engineer is required to develop a friendly relationship with the secondary target (public safety office) who has authority to grant access to the laboratory facility. Intent is to develop trust between the social engineer and secondary target.

Step V: Exploit Relationship

I. Priming the Target: Social engineer who is impersonating the tertiary target's (faculty member) student research assistant urgently needs

access to the server room to conduct research calculations to meet deadlines with contracted companies.

II. **Elicitation:** Social engineer explains to the secondary target (public safety officer) that he/she forgot his/her access card to the server room across campus (in their office). It would take too long and inconvenient for the tertiary target's (faculty member) student research assistant to go back and grab the access card.

Step VI:

I. **Maintenance:** After the social engineer has successfully gained access to the unauthorized server room. The tertiary target's (faculty member) student research assistant generously thanks the secondary target (public safety officer) for his/her ability to assist in authorizing the secondary target's access.

II. **Transition:** Since the social engineer was able to gain access to the unauthorized server room, the social engineer can now proceed to the "Goal Satisfaction" stage.

III. **Goal Satisfaction:** SE has accomplished his/her original goal of gaining unauthorized access to the institution's laboratory facility.

E. Command and Control Framework

The author proposes the design and development of a command-and-control application programming interface framework from which to facilitate the attack of the project's bots/botnets. As seen in figure four, the author developed the initial logical flow of the command-and-control application programming interface. As we step through the logic of the command-and-control application programming interface, the author will be developing automation capabilities for all of the functions proposed. The application programming interface will create all bot identities based on a secret key and value pair. All identities will be posted to an internal create, read, update, and delete database. Once bot identity is created, it will be automatically pushed to a publicly facing Git repository. This repository can either be GitHub or GitLab. In parallel, the application programming interface will be generating malicious attack plan association with the newly created bot identity. These attack plans are pulled from an existing attack dataset released for open intelligence. Once the automation of the administrative work is complete, the application programming interface will sit and wait for bots to phone home to initiate a handshake. If no bots have phoned home, the application programming interface will continue to sit there waiting. If the bot has phoned home and is the initial handshake, the application programming interface will associate their identity to the attack plan in the create, read, update, delete database, and send those attack plans back to the malicious bot. If it is not the first handshake and the malicious bot needs to post data to the specified create, read, update database, it will do so; otherwise, the application will continue to wait for the next communications with the bot.

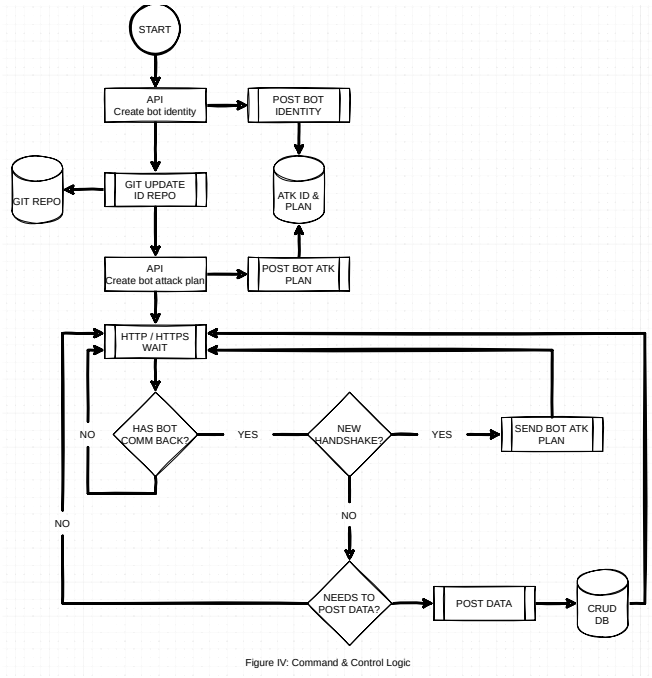

Figure IV: Command & Control Logic

IV. PRELIMINARY / EXPECTED RESULTS

A. Preliminary

As outlined in this paper's Background & Related Work's API section, the author analyzed two of the current leading Python REST API Frameworks, and a new modern API centered framework. During first round of evaluations of Django's REST capabilities, it was apparent Django would not be a feasible framework to develop the features the author proposed in their Proposed Work section [9, 10]. The author acknowledges the powerful capabilities that Django provides to developers. The primary justifications to not develop with Django is its massive code library that needed to be imported and its substantial lines of code needed to initiate a skeletal API endpoint [9, 10]. In the next round of evaluations of Python RESTFUL frameworks, the author evaluated Flask [11 – 16]. Flask had better feature and capabilities in API development to Django and showed great promise. This was due to the innate nature of Flask as being a micro-framework [11, 12, 16]. The author found that the lightweight skeletal Flask framework and substantial modules to extend its capabilities would allow for fast development and maintenance. This is exemplified by the developers of the Flask-RESTful stating, "Flask-RESTful is an extension for Flask that adds support for quickly building REST APIs. It is a lightweight abstraction that works with your existing ORM/libraries. Flask-RESTful encourages best practices with minimal setup. If you are familiar with Flask, Flask-RESTful should be easy to pick up" [12]. In comparison to Django, Flask outclassed Django in the number of lines of code needed to have a fully functioning API endpoint up and running. While CH4RL3SCH4L3M4GN3 is currently in the early stages of development utilizing Flask-RESTful, the author is also considering adopting the latest modern Python API framework FastAPI [17 – 19]. This modern API focused framework was recently released in the Python Package Index on 07 Oct 2021. It has a vibrant community and multiple applications and sponsorships including but not limited to ML/AI, Neural SaaS, and FinTech. The author is heavily

considering utilizing this modern API framework as the architecture of CH4RL3SCH4L3M4GN3. More analysis and testing need to be done before final adoption.

In regard to bot/botnet development, the author has scripted a simple Python script that utilizes the built-in requests module to facilitate communications to remote command and control API. The author is currently considering design ideas based on the research on bots/botnets [20 – 28]. This leads to the malware capabilities development status, which will heavily influence the design development of the malicious bot. The author chose to develop two malware capabilities that encrypt data [34 – 36] and exfiltrate data [37 – 40]. The author will be focused primarily on symmetric encryption as the cryptographic scheme. The author will develop a simple data exfiltration over HTTP/HTTPS traffic via API CRUD operations.

B. *Expected Results*

1) *Primary Plan*

The author's primary plan is to have fully functioning attack simulations designed and developed by the end of the project's timeline. The simulations will be based on the social engineer attack scenarios proposed in the author's Proposed Work, Social Engineering Scenario section above. The attack plans carried out by the malicious bots will be based on figure 2a, 2b, 3a, and 3b. The malicious bots and attack plans will be facilitated by the command-and-control API, see figure 4.

2) *Fallback Plan*

The author's fallback plan if the author's primary plan exceeds the timeline of the project is to establish HTTP/HTTPS communications between the malicious bot and the command-and-control API.

3) *Last Resort Plan*

The author's last and final plan is to run any code developed and if it runs, it runs. There is an implicit expectation that the malicious bot will not be able to communicate to the command-and-control API, and API functions are either partially developed or still needs to be developed.

V. CONCLUSION & FUTURE WORK

A. *Conclusion*

In conclusion, the author provided a brief introduction to the current trends of bots/botnets, as well as the command-and-control tactics, techniques, and procedures used by APT groups to carry out cyber-based attacks. The author also provided an overview of the technologies being used for the development of the command-and-control microservices API: CH4RL3SCH4L3M4GN3. The author's proposed work provides a high-level view of the design, logic, and capabilities of this project, but please not that these proposals are subject to change. Finally, the author outlined the preliminary work conducted as of 10 Dec 2021, with three expected results of this project. The author chose to account for three possibilities due to the vast technical skills required to design and develop this project compared to the author's current technical skills. The author's primary plan is the best possible outcome for the project, the fallback plan will have some technical implementations but not all, and the last-resort plan will have minimal implementation and will only provide development code.

B. *Future Work*

In the author's future work, depending on the outcome of this project, the author plans to bring this project from lab simulation to live environment deployment. This will require the acquisition of a public domain and server to host the project API. The author realizes that there needs to be quantitative data aggregation on the performance of the API and malicious bots. The author will be designing and developing a system to achieve this goal.

ACKNOWLEDGMENT

The author thanks the contributions of the MITRE Foundation for their extensive work on the ATT&CK Framework, Roy T. Fielding for his dissertation in proposing the concept of Representational State Transfer (REST) Architecture, Sebastián Ramírez for his valuable work on the creation of FastAPI and finally the School of Computer Science and Engineering, Jack Welch College of Business and Technology, Sacred Heart University for continuing to give me inspirations for higher education social engineering attacks. Without their contributions and influence this project would not have been possible.

REFERENCES

[1] M. Lee and D. Lewis, "Clustering Disparate Attacks: Mapping the Activities of the Advanced Persistent Threat", Virus Bulletin Conference, 2011.

[2] I. Ghafir and V. Prenosil, "Advanced Persistent Threat Attack Detection: An Overview", International Journal of Advancements in Computer Networks and Its Security-IJCNS, Vol. 4: Issue 4, pp.50–54, 2014.

[3] P. Chen, L. Desmet, and C. Huygens, "A Study on Advanced Persistent Threats", IFIP International Federation for Information Processing 2014, D. De Decker and A. Zuguete: CMS 2014, LNCS 8735, pp.63–72, 2014.

[4] P. Hu, H. Li, H. Fu, D. Cansever, and P. Mohapatra, "Dynamic Defense Strategy against Advanced Persistent Threat with Insiders", 2015 IEEE Conference on Computer Communications (INFOCOM), IEEE, pp.747–755, 2015.

[5] A. Lemay, J. Calvet, F. Menet, and J. Fernandez, "Survey of Publicly Available Reports on Advanced Persistent Threat Actors", Computers & Security, 2017. http://dx.doi.org/doi: 10.1016/j.cose.2017.08.005.

[6] J. Chen, C. Su, K. Yeh, and M. Yung, "Special Issue of Advanced Persistent Threat", Future Generation Computer Systems, Vol. 79, pp.243–246, 2018.

[7] O. Adelaiye, A. Ajibola, and S. Faki, "Evaluating Advanced Persistent Threats Mitigation Effects: A Review", International Journal of Information Security Science, Vol. 7, No. 4, pp.159–171, 2019.

[8] C. Chen, H. Lai, and D. (Marian) Wen, "Evolution of Advanced Persistent Threat (APT) Attacks and Actors", ICS 2018, CCIS 1013, pp.76–81, 2019.

[9] D.S. Foundation, Django Documentation: Release 3.2.10.dev, 2021, pp.687–1577.

[10] D. Rubio, "Beginning Django, Chapter 12: REST Services with Django", 2017, pp.549–566.

[11] P. Projects, Flask Documentation (1.1.x): Release 1.1.4, 2021.

[12] K. Conroy, R. Horn, F. Stratton, Flask-RESTful Documentation: Release 0.3.8, 2020.

[13] P. Projects, Jinja Documentation (3.0.x): Release 3.0.0, 2021.

[14] P. Projects, Werkzeug Documentation (1.0.x): Release 1.0.1, 2021.

[15] K. Relan, “Building REST APIs with Flask: Create Python Web Services with MySQL”, Apress, 2019.

[16] A. M. Bekabil, “REST API Implementation with Flask-Python", LAPIN AMK, Lapland University of Applied Sciences, 2014.

[17] S. Ramirez, FastAPI, 2021. [Online]. Available: https://fastapi.tiangolo.com. [Accessed: 01–Dec–2021].

[18] T. Christie, Starlette, 2021. [Online]. Available: https://www.starlette.io/. [Accessed: 05–Dec–2021].

[19] S. Colvin, pydantic, 2021. [Online]. Available: https://pydantic-docs.helpmanual.io/. [Accessed: 07–Dec–2021]

[20] P. Wang, B. Aslam, and C. Zou, “Peer-to-Peer Botnets”, Peter Stavroulakis, Mark Stamp (Eds.), Handbook of Information and Communication Security, pp.335–350, 2010.

[21] N. Hachem, Y. Mustapha, G. Granadillo, and H. Debar, “Botnets: Lifecycle and Taxonomy”, Institute of Electrical and Electronics Engineers (IEEE), 2011.

[22] S. Silva, R. Silva, R. Pinto, amd R. Salles, “Botnets: A Survey”, Computer Networks 57, pp.378–403, 2013.

[23] M. Rahimipour and S. Jamali, “A Survey on Botnets and Web-based Botnet Characteristics”, International Journal of Computer Science Engineering and Technology (IJCSET), Nov. 2014, Vol. 4, Issue 11, pp.282–286.

[24] S. Khattak, N. Ramay, K. Khan, A. Syed, and S. Khayam, “A Taxonomy of Botnet Behavior, Detection, and Defense”, IEEE Communications Survey & Tutorials, Vol. 16, No. 2, Second Quarter 2014, pp.898–924, 2014.

[25] M. Eslahi, M.S. Rohmad, H. Nilsaz, M. Naseri, N.M. Tahir, and H. Hashin, “Periodicity Classification of HTTP Traffic to Detect HTTP Botnets”, Institute of Electrical and Electronics Engineers (IEEE), 2015.

[26] A. Lemay, J. Fernandez, and S. Knight, “A Modbus Command and Control Chnnel”, Institute of Electrical and Electronics Engineers (IEEE), 2016.

[27] J. Zhang, R. Zhang, Y. Zhang, and G. Yan, “The Rise of Social Botnets: Attacks and Countermeasures”, Institute of Electrical and Electronics Engineers (IEEE), 2016.

[28] Z. Nafarieh, E. Mahdipour, and H. Javadi, “Detecting Bot Networks Based On HTTP and TLS Traffic Analysis”, J. ADV COMP ENG TECHNOL, 6(2) Spring 2020, pp.61–70, 2020.

[29] J. Gardiner, M. Cova, and S. Nagaraja, “Command & Control: Understanding, Denying and Detecting”, University of Birmingham, 2014.

[30] P. Stodola and J. Mazal, “Architecture of the Advanced Command and Control System”, 2017 International Conference on Military Technologies (ICMT), pp.340–343, 2017.

[31] J. Piet, B. Anderson, and D. McGrew, “An In-Depth Study of Open-Source Command and Control Frameworks”, 2018 13th International Conference on Malicious and Unwanted Software: “Know Your Enemy” (MALWARE), pp.23–30, 2018.

[32] M. Baden, C. Torres, B. Pontiveros, and R. State, “Whispering Botnet Command and Control Instructions”, 2019 Crypto Valley Conference on Blockchain Technology (CVCBT), pp.77–81, 2019.

[33] F. Sadique and S. Sengupta, “Analysis of Attacker Behavior in Compromised Hosts During Command and Control”, ICC 2021 - IEEE International Conference on Communications, 2021.

[34] J. Laser and V. Jain, “A Comparative Survey of Various Cryptographic Techniques”, International Research Journal of Engineering and Technology (IRJET), Vol. 3, Issue: 03, pp.162–171, 2016.

[35] M. Bokhari and Q. Shallal, “A Review on Symmetric Key Encryption Techniques in Cryptography”, International Journal of Computer Applications (0975 – 8887), Vol. 147, No. 10, pp.43–48, 2016.

[36] P. Matta, M. Arora, and D. Sharma, “A Comparative Curvey on Data Encryption Techniques: Big Data Perspective”, Materials Today: Proceedings, Vol. 46, Part 20, pp.11035–11039, 2021.

[37] P. Nyakomitta and S. Abeka, “A Survey of Data Exfiltration Prevention Techniques”, First International Conference on Computer Engineering, International Journal of Scientific Research in Science and Technology Vol. 5, Issue 8, pp.45–52, 2020.

[38] A. Nadler, A. Aminov, and A. Shabtai, “Dection of Malicious and Low Throughput Data Exfiltration Over the DNS Protocol”, Computer & Security (2018), doi: https://doi.org/10.1016/j.cose.2018.09.006

[39] J. Steadman and S. Hayward, “DNSxD: Detecting Data Exfiltration Over DNS”, 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2018.

[40] L. Liu, O. Vel, Q. Han, J. Zhang, and Y. Xiang, “Detecting and Preventing Cyber Insider Threats: A Survey”, IEEE Communications Survey & Tutorials, pp.1–21, 2018.

[41] K. Ivaturi, L. Janczewski, “A Taxonomy for Social Engineering Attacks”, Association for Information Systems: AIS Electronic Library (AISeL), CONF-IRM 2011 Proceedings. 15. 2011.

[42] F. Mouton, L. Leene, M. M. Malan and H. S. Venter. Towards an Ontological Model Defining the Social Engineering Domain, in: K.K. Kimppa et al. (Eds.): HCC11 2014, IFIP AICT 431, 2014, pp.266–279.

[43] F. Mouton, L. Leene, and H.S. Venter. Social Engineering Attack Detection Model: SEADMv2, 2015 International Conference on Cyberworlds, Institute of Electrical and Electronics Engineers (IEEE), 2015, pp.216–223.

[44] F. Mouton, L. Leene, M.M. Malan and H.S. Venter. Social Engineering Attack Example, Templates and Scenarios, Computer & Security, Volume 59, 2016, pp.186–209.

[45] T. Nguyen, S. Bhatia, “Higher Education Social Engineering Attack Scenario, Awareness & Training Model”, Journal of the Colloquium for Information Systems Security Education (CISSE), Vol. 8, No. 1, Dec. 2020