DeadDrop

a clever command and control framework for penetration testers

Project Proposal/Concept (P1)

Jann Arellano, Brian Buslon, Keaton Clark, Lloyd Gonzales

CS 425, Fall 2023

Instructors: Sara Davis, Devrin Lee, Vinh Le

Advisor: Shamik Sengupta

Department of Computer Science and Engineering
University of Nevada, Reno

October 16, 2023

Table of Contents

Abstract

We propose DeadDrop, a command and control (C2) framework used in post-exploitation activities and penetration testing to create malware payloads, manage compromised devices, and generate operational reports. Unlike existing frameworks that communicate directly with attacker domains, DeadDrop focuses on leveraging features in legitimate websites such as YouTube and Wikipedia to communicate with devices and exfiltrate data, masking its activity within the noise of popular websites.

The novelty of this approach is valuable to offensive and defensive cybersecurity professionals alike, as corporate networks and firewalls often "trust" the traffic of large, well-known websites. By abusing this trust, DeadDrop provides security engineers with new insight into identifying covert attacker techniques and security weaknesses. For this reason, we believe that DeadDrop is a competitive product in a saturated market of other C2 frameworks and is a significant contribution to cybersecurity as a whole.

I. Project Description

This section is around 1900 words. We recognize this is long (and will likely cost some points), but we would rather be detailed in our proposal than incomplete.

Goals, Objectives, and Significance

At a high level, DeadDrop's primary objective is to provide cybersecurity professionals with a novel form of covert communication between a target network and an attacker over legitimate websites, while still providing the features and services that penetration testers (the "red team") expect out of existing command and control (C2) frameworks.

DeadDrop is significant in that it encourages cybersecurity professionals to take different approaches to identifying C2-related traffic. Existing methods for "categorizing" C2 traffic often depend on looking for intermittent, anomalous traffic to newly created domains. With adequate network logging and rapid alerting, compromised machines directly communicating with C2 servers can easily be identified from typical traffic. However, this can be defeated by using popular websites such as YouTube or Wikipedia as a form of communication instead; for example, commands and responses might be "sent" by editing a Wikipedia user page, while large files might be exfiltrated from company networks as an encoded YouTube video.

In turn, DeadDrop's true "goal" is to increase awareness of covert communication streams that could be used by highly skilled threat actors. To achieve this, DeadDrop provides organizations with the tools needed to develop *and* test their defenses against these techniques through a highly configurable and extensible framework.

Functionality and Requirements

DeadDrop aims to include all of the functionality of existing C2 frameworks, including reporting and malware generation. It also implements novel communication protocols for compromised devices connected to the Internet, leveraging services on legitimate websites to indirectly communicate with devices (similar to the espionage practice of using dead drops).

This functionality is best implemented as three modular segments, each of which can be developed in parallel and provides specific functionality to the framework.

First, the **C2 server** must provide a *team* of attackers with a user-friendly interface to interact with compromised devices, generate new payloads, and log all actions taken. Functionally, the server should:

- send commands to and interpret responses from compromised devices, possibly over many different protocols;
- onboard newly compromised devices and ensure these devices (and any other incoming traffic) were authorized by the attackers;
- reformat raw output for common responses, such as directory listings or ARP tables, into convenient views such as sortable tables and graphs;
- log all communications and actions between the C2 server and its agents, providing the ability to export these raw logs (as JSON/XML/etc.) and a pre-formatted HTML report;
- and provide a user authentication system for interacting with the framework, ensuring accountability for actions taken by individual users.

Second, the **C2 agents** must provide the attacker with the means to interact with the computer the agent/malware is installed on. Functionally, agents should:

- support one or more communication protocols, providing fallback options when contact is lost over any single protocol;
- at minimum, provide a mechanism to execute commands received over the communication protocol and return the output in a standard message format
- preferably, provide built-in "commands" for common actions such as device discovery or filesystem enumeration that yield well-formatted responses for easier analysis and presentation by the C2 server's frontend;
- survive a restart of the machine (persistence) *if time allows*;
- and evade antivirus detection through polymorphic code *if time allows*.

Finally, the **communication protocols** are the abstract definitions and formal implementations of our novel idea. Functionally, communication protocols should:

- be *independent* of any particular agent; that is, it should serve as an opaque layer that can be seen by any agent as a very delayed but reliable form of TCP, and does not depend on a particular agent's features to operate;
- support end-to-end encryption and define a procedure for coordinating the encryption key between the server and the agent;

- and define a procedure coordinating the website and specific service used, as well as how to handle delays, out-of-order messages, or lost messages; for example, if YouTube is used as a communication channel, the server might "drop" new commands as comments on a video uploaded to a specific channel agreed upon by the server and agent ahead of time.

Audience and Purpose

DeadDrop's primary purpose is to aid legitimate security professionals (the "red team") in an activity known as penetration testing, which assesses the company's ability to defend against attacks and protect its data. These penetration testers may be in-house offensive teams at large organizations or external firms.

In a penetration test, the red team identifies vulnerabilities in an organization's devices, using these vulnerabilities to install malware that allows them to control that device. By leveraging the "trust" of this compromised device, the red team can then plant malware on more valuable devices with sensitive information (some of which may not be directly connected to the public internet), with the goal of exfiltrating this sensitive data out of the network without being caught.

Penetration tests are often coordinated using C2 frameworks, which create individual malware payloads, communicate with compromised devices, and keep track of all data received and any actions taken. Both the use of C2 frameworks and penetration testing as a whole greatly benefit security engineers (the "blue team"), as they can identify holes in detecting malicious activity through the reporting and logging functionality provided by the C2 framework.

In turn, DeadDrop's primary audience is cybersecurity professionals, including penetration testers and defense analysts. Organizations can use the information generated by DeadDrop to monitor C2 activity from different access points and perspectives. As penetration testers move throughout the organization's network, this provides the organization with the logs and operational reports needed to patch vulnerabilities and gaps before a real data breach happens.

In short: DeadDrop provides organizations with the opportunity to simulate an attacker and proactively defend against (our) novel attack techniques.

Anticipated Challenges

We anticipate four major challenges throughout our project:

- **Regulatory and compliance:** We anticipate regulatory issues to be the largest challenge with this project. As with any activity that deviates from a service's intended use, we must avoid violating the terms of service for the websites we intend to use as part of our covert communication methods. Our primary plan to address this is to set up mock services that emulate the functionality of the true websites when testing and demonstrating DeadDrop. Furthermore, as is consistent with industry practices for penetration testing, we will enforce a logging policy to ensure any actions we take (whether on our own networks or on an external network) can be undone if necessary.

- **Technical hurdles:** This project involves several networking and low-level challenges for which we have little or no experience, namely malware design, communication networks, communication protocols, and penetration testing. Over the course of this project, we will be getting hands-on experience with these topics and will refer to our advisor as needed.
- **Timelines and scheduling:** We have found that our team is rarely available for full team meetings due to classes, work, and personal activities. As a result, most work must be done remotely and asynchronously. This can make collaborative work difficult and is particularly problematic for components that strongly depend on collaborative work. For example, frontend progress might depend on knowledge from the backend developers (who may not be available at the same time as frontend developers). While we will make every effort necessary to prepare our schedule for team meetings, we do expect asynchronous work to be a challenge.
- **Scope and resource constraints:** While we have made every effort possible to determine the resources required to complete this project (as described in the *Budget and Budget Justification* section), the scope necessary to effectively test and demonstrate DeadDrop will determine both the physical and cloud resources needed. As development progresses, we anticipate that resource requirements for testing will increase, requiring us to determine suitable scopes in advance.

Future Development

The absolute minimum our project must provide is a modular framework that can support the full life cycle of a penetration test and implements our novel communication protocols. This means that various convenience features (such as PDF report generation) or malware functionality (such as antivirus evasion and persistence) can be missing from the final result without harming DeadDrop's significance or usability. This is often the case with other open-source frameworks, which do not provide any built-in functionality for antivirus evasion.

As a result, we believe that a major opportunity for future development lies in payload construction; the most basic C2 agent we can develop is a simple Python script that is platform-independent. As this project progresses (including past the end of the academic year), it may make sense to bundle DeadDrop with automated and custom-built solutions for obfuscating payloads against antivirus solutions and supporting persistence.

Additionally, because DeadDrop is inherently designed to be modular, additional agents or communication protocols could be implemented and included with DeadDrop that cover new techniques discovered "in the wild" or described in academic papers, which allows organizations to continuously improve their defenses against new threats by extending DeadDrop.

Technology Stack

The project can broadly be divided into three modular segments (as described in the *Functionality and Requirements* section), each of which depends on its own technology stack. The libraries, programming languages, and technologies shown below are what we *expect* to use.

For the *C2 server*:

- **Svelte (JavaScript)**: for building the frontend
- **Django (Python):** for building the backend
- **SQLite:** underlying database connected to Django through Django's ORM

For the C2 agents (and by extension, communication protocol implementation):

- **Docker Compose:** for compiling and testing platform-specific malware
- **Python, C, C++:** for implementing standard agent functionality and specific communication protocols
- **x86 assembly:** where needed, for handwritten antivirus evasion routines
- **PyInstaller:** for packaging Python implementations of agents with dependencies
- **ffmpeg:** for audio and video processing in one of our planned communication protocols

Additionally, we plan on using the following technologies for project management, testing, DevOps, and other overhead not strictly related to implementation:

- **Microsoft Azure/AWS:** for hosting cloned/look-alike instances of Wikipedia, YouTube, etc; may also be used to create a virtual network for testing and demonstration
- **black:** for Python code formatting and PEP8 style guidance
- **mypy:** for static Python code analysis
- **pytest:** for unit and integration testing
- **tox:** for automated Python builds and testing
- **GitHub issues/projects/wiki:** for overall project management

Team, Advisory, and Professional Overview

DeadDrop is built by a diverse team of students within the Computer Science and Engineering department.

- **Jann Arellano:** Jann is experienced in web development with Svelte and backend experience with FastAPI and Flask. His interests include full-stack development and data science. He will be involved in the backend and frontend development for the C2 server.
- **Brian Buslon:** Brian is experienced in web development with Django, Flask, and React/React Native through the Expo framework and is familiar with PostgreSQL. He is interested in both frontend development and projects related to machine learning. He will be involved in the backend and frontend development of the C2 server.
- **Keaton Clark:** Keaton is a firmware engineer who is experienced in RTOS, Linux, and various bare-metal platforms. He will be providing build chains, system integration, and backend core logic, as well as developing the communication protocol between the backend and C2 agents.

- **Lloyd Gonzales:** Lloyd has experience in data analytics in Python and Matlab, as well as offensive cybersecurity experience from internships and national competitions. As part of his interest in a career in penetration testing, he will be responsible for developing DeadDrop's C2 agents and implementing communication protocols.

Our current external advisor is Shamik Sengupta, a professor within the Computer Science and Engineering department with extensive experience in cybersecurity projects and research. We believe that his experience will help us greatly in guiding our development and identifying opportunities to align DeadDrop with industry practices.

Although each of us has distinct career interests, the modularity of the project (as described in the *Functionality and Requirements* section) provides each of us with the opportunity to explore industry techniques and libraries related to our professional goals and demonstrate proficiency in technologies sought by employers. Furthermore, we hope to develop a research paper from our work, demonstrating our communication and research skills to employers and academia.

II. Market Potential and Open Source Significance

Market Analysis

From a market potential perspective, DeadDrop has great potential because it uses trusted services to exploit vulnerabilities. To our knowledge, no major C2 frameworks leverage man-in-the-middle attacks. Existing C2 frameworks like Mythic use a more direct approach and are often easier to detect.

Commercially, DeadDrop would be sought out by large organizations with in-house red teams like MGM Resorts and penetration testing firms like Mandiant that directly benefit from the use of a C2 framework [1]. DeadDrop is particularly valuable for its novel attack approach that no other C2 framework currently provides, allowing these firms to find new ways to improve organizational defenses.

From an open-source perspective, DeadDrop supports community contributions for both agents and communication protocols through its modular design. Additionally, web developers and other industry professionals add missing reporting functionality. Because of this flexibility, we can reasonably expect interest from users and companies alike that may want to adapt DeadDrop to evaluate their own environment's defenses.

Competitive Analysis

There are two major competitors to DeadDrop: Cobalt Strike (proprietary) and Mythic (open-source).

Cobalt Strike emulates advanced persistent threats (APTs) to assess system security. Its core component, the "Beacon" payload, manages command and control, data exfiltration, and lateral movement using a multi-stage payload system to bypass detection [2]. In addition to coordinating compromised systems, Cobalt Strike also supports spear phishing and report generation for security assessments. Ethical hackers, cybersecurity professionals, and red teamers utilize it to evaluate security, find vulnerabilities, and enhance cybersecurity. It is also used for training and helping professionals understand real-world attack techniques.

Mythic is an open-source C2 framework focused on red teaming operations, offering a user-friendly web interface and quality-of-life features for managing and controlling agents on infiltrated systems [3]. Mythic also mimics real-world infiltration techniques, providing a Jupyter scripting engine, dedicated tasking services, and payload generation for Windows, macOS, and Linux [4]. Similar to Cobalt Strike and other C2 frameworks, Mythic is used to evaluate and enhance organizational defenses.

Although Cobalt Strike and Mythic are two of the largest C2 frameworks, additional alternatives include Empire, Koadic, Covenant, Sliver, and SilentTrinity. These range greatly in maturity, ranging from proof of concept scripts to "full" modular frameworks.

Competitive Advantage

While DeadDrop is unlikely to have full feature parity by the end of the year compared to alternative products, especially with respect to payload features (such as antivirus evasion), we believe that DeadDrop's novel communication protocol provides it a large edge over existing products. In all branches of security, the most dangerous threats are those that you are not aware of and cannot emulate yourself. Although there are no similar large-scale projects to DeadDrop, this does not mean these techniques (or similar techniques) are not already in use; in turn, we believe that many organizations industry will adopt DeadDrop in testing and developing new defenses.

Our intent to make this project open-source is a proactive approach that benefits the broader cybersecurity community. It involves sharing knowledge, fostering community involvement, enabling rapid response to threats, and offering customization. This approach also promotes transparency, cost savings, global collaboration, and education for security professionals. This contrasts with the drawbacks of keeping the code closed-source, which makes penetration testing inaccessible to small organizations but accessible to those who are willing to pirate - a common complaint with competitors such as Cobalt Strike.

DeadDrop ultimately enhances network security while contributing to the collective defense against evolving threats, doing so in a competitive manner.

References

[1] Lucian Constantin, "Here is why you should have Cobalt Strike detection in place," CSO Online. Accessed: Oct. 16, 2023. [Online]. Available: https://www.csoonline.com/article/574143/here-is-why-you-should-have-cobalt-strike-detection-in-place.html

[2] Fortra, "Features | Beacon, C2 Profiles, Attack Packages, and More," Cobalt Strike. Accessed: Oct. 16, 2023. [Online]. Available: https://www.cobaltstrike.com/product/features

[3] Justin Palk, "Introduction to Mythic C2." Accessed: Oct. 16, 2023. [Online]. Available: https://redsiege.com/blog/2023/06/introduction-to-mythic-c2/

[4] Mythic Developers, "Mythic Documentation." Accessed: Oct. 16, 2023. [Online]. Available: https://docs.mythic-c2.net/

III. Time Worked on Project Concept

Note that the recorded time includes time spent writing and formatting these sections in addition to learning about cybersecurity topics related to the project.

Team Member	Hours	Sections Contributed
Jann Arellano	6	- Project Description - Audience and Purpose - Technology Stack - Team, Advisory, and Professional Overview - Market Potential and Open Source Significance - Market Analysis
Brian Buslon	5	- Project Description - Future Development - Team, Advisory, and Professional Overview - Market Potential and Open Source Significance - Competitive Analysis
Keaton Clark	6	- Project Description - Anticipated Challenges - Technology Stack - Functionality and Requirements - Team, Advisory, and Professional Overview - Market Potential and Open Source Significance - Competitive Advantage
Lloyd Gonzales	9	- Abstract - Project Description - Goals, Objectives, and Significance - Functionality and Requirements - Audience and Purpose - Budget and Budget Justification - Project Related Resources - (general formatting and organization)

IV. Budget and Budget Justification

We do not anticipate any *required* purchases for this project's development, testing, and demonstration. However, emulating the network of a relatively small organization with multiple subnets (some of which may not be connected to the internet, and some of which may only be accessible with specific machines) that allows us to test and demonstrate DeadDrop is best achieved through cloud services. We believe that allocating funding for a virtual private cloud, such as one hosted on AWS or Azure, would be beneficial to the project's development when our own physical machines or a set of local VMs are not sufficient. Using cloud services allows us to switch to different host operating systems or quickly change the network architecture on the fly, which is more difficult with physical systems or self-hosted VMs.

A realistic application of DeadDrop might be performing a penetration test against the virtual private network described by Microsoft in its virtual network guide and mesh network guide for Azure, in which a single internet-connected machine is connected to multiple subnets that host more sensitive information. This is reflective of real-world network architectures in which high-value machines are not connected to the internet; in turn, a C2 framework *should* support forwarding messages between C2 agents in such an environment.

On Azure, VMs with a reasonable amount of computing power (~4 GB RAM and 8 GB storage) cost $30/month to maintain, whether they're using Windows or a Linux distro. Such a machine would be reflective of a "real" machine's computing power and is required for high-intensity tasks (like uploading/encoding videos to the internet). VMs with very little computing power (say, if we wanted to store a small database) cost about $8/month to maintain. This does not include some overhead associated with the virtual network, which is not free.

Within reason, a very small network of 8 "weak" machines (either Windows or Linux) in multiple private subnets, in addition to a more powerful internet-facing host and a private virtual network, will cost about $100 a month according to Azure's price estimates. This is an *overestimate* of the number of machines and computing power that would be necessary to test the functionality of DeadDrop, as we would likely deallocate these machines during development periods when our own machines are sufficient for testing.

In turn, we believe that with a combination of Azure student credit and up to **$400 of additional credit** - equivalent to four months of "large-scale" testing, for one semester - we would have the resources necessary to complete the project, with a significant safety margin. The student credit would be used *before* any purchases with University-provided credit are made.

V. Project Related Resources

Problem Domain Book

Learn Penetration Testing is an introductory book to both the technical and non-technical principles of penetration testing. It covers not only the specific techniques that attackers use to identify and exploit vulnerabilities, but also the overall "life cycle" of an attack. This is particularly important; although the novelty of this project comes from its technical implementation of a covert communication protocol, the project as a whole is only useful if it can be used to orchestrate an attack from start to finish.

This book is particularly helpful as it identifies specific tasks that penetration testers perform, including requirements for reporting and acting on findings. Any other ideal "problem-domain" book would cover the penetration testing process in-depth, including all non-technical organization-specific topics, such that we can model our framework around the needs of an attacker following that process.

Reference Articles

Each of the publications listed below detail penetration testing at the organizational level; that is, rather than focus on specific technical approaches, these articles detail the general procedures and ethics that teams follow when conducting penetration tests in a variety of different environments. Additionally, these articles also discuss the general impact of penetration testing from a governance and risk perspective, as well as the "human" aspect of cybersecurity (such as social engineering).

- *An overview of vulnerability assessment and penetration testing techniques*
- *A study on penetration testing process and tools*
- *MTC2: A command and control framework for moving target defense and cyber resilience*

Additional Resources

- The Mythic C2 documentation details payload and agent development for the Mythic C2 framework, which details Mythic's requirements for what new agents must implement. This is particularly helpful, as it helps inform us on what data is needed for communication between the server and multiple agents, as well as what their message format looks like. It also contains an extensive amount of general documentation on Mythic's architecture, components, and configuration..
- A list of various open-source C2 frameworks is available on GitHub, ranging from fully-fledged frameworks (like Mythic) to proofs of concept (usually just two scripts, one for the server and another for the agent). This allows us to evaluate what functionality is common among many different frameworks (and therefore desirable), as well as how other well-designed frameworks accept new functionality.
- The Sliver C2 documentation details both the architecture of Sliver as well as various other resources related to implementing specific penetration testing techniques (such as antivirus evasion) that are not covered in the Mythic documentation. This includes links to the Veil Framework and PEzor, both of which are automated solutions to developing payloads that evade common antivirus solutions.

- A list of resources for implementing malware persistence (that is, malware that survives a restart of the computer) and detecting and analyzing persistent malware techniques, is available on GitHub, along with a general overview of how persistence techniques work. This can be particularly helpful in implementing our C2 agents in a “real world” context; although VMs in the cloud might not restart often, real user machines do.