DeadDrop

a clever command and control framework for penetration testers

Progress Demo (P4)

Jann Arellano, Brian Buslon, Keaton Clark, Lloyd Gonzales
Team 08

Advisor:
Shamik Sengupta
Professor
University of Nevada, Reno

CS 426, Spring 2024
Instructors: David Feil-Seifer, Devrin Lee, Sara Davis

Department of Computer Science and Engineering
University of Nevada, Reno
March 11, 2024

Table of Contents

I. Project Status

At the fall prototype demo, we had successfully developed a proof of concept in which we successfully sent arbitrary data over YouTube and used it to execute commands remotely on an "infected" device. Our challenge this semester was to convert this prototype into something actually usable from an industry perspective, providing the foundation for other users to develop an ecosystem of agents and protocols built around convert communication.

Since then, we have made extensive progress toward building this foundation – although many of these changes are invisible and have yet to manifest as part of the web interface, the functionality exists and is accessible through internal interfaces. (That is, the functionality is exposed through our RESTful API and internal libraries, but can't be invoked through UI elements on the frontend just yet.)

A brief summary of major accomplishments includes:

- The implementation of a package manager, allowing users to build, distribute, and install multiple agents in individual DeadDrop installations;
- The development of standardized metadata files that allow agents to expose important interfaces and information in a language-agnostic manner;
- The publishing of a meta-library for DeadDrop, providing common interfaces and expectations for agents, messages, and other core data structures;
- Continued research and development toward adapting the YouTube communication protocol to operate over live Zoom calls, in addition to a new architecture and a mock YouTube service;
- New research towards operating a communication protocol over Craigslist and Facebook Marketplace;
- Continued progress toward the authentication and user system, allowing proper logging and accountability for actions taken through the framework; and
- The use of Docker Compose on all major components (the server, the frontend, and Pygin), which reduces setup time on individual machines and increases overall reliability.

At this point in time, much of our remaining work is in implementing more tightly scoped features, such as encryption and message signing over individual protocols, handling logs generated throughout the framework, and rendering high-level statistics. These provide functionality that is more expected of industry-grade tools, allowing us to demonstrate the flexibility of the framework as a whole.

We are confident that the minimum for this to be considered a usable C2 framework (by security and networking professionals) can be completed by Spring Break at the latest, with more "flashy" functionality – such as stealing Discord sessions over our protocols in a live demo – achievable by mid-April.

II. Implemented Requirements

The following section describes *every* individual functional and non-functional requirement that has been specified throughout the PA documents, including those that we believe are unlikely to be completed this semester.

- Level 1 requirements are those that we are confident will be done before Spring Break.
- Level 2 requirements are those that we are confident will be done before the final demo.
- Level 3 requirements are those that are nice (or necessary) in industry, but are more uncertain in terms of a timeline.

Much of the work leading up to the progress demo has been foundational, providing the libraries and functionality on the backend needed to make each of these individual features possible. In turn, the majority of these requirements are tightly-scoped features that build upon this foundation and have yet to be implemented in *full* (i.e. correspond to a button on the frontend), but either already exist internally or can be quickly developed with this new foundation. To be more precise, the majority of the requirements in Planned Requirements can be fulfilled with about four hours of additional work each.

The project board at https://github.com/orgs/unr-deaddrop/projects/2 is a better overview of our planning of individual work items. Many of the requirements below correspond one-to-one to specific GitHub issues, but many lower-level GitHub issues are absent from the list below. 62 issues have been closed since the prototype demo, with about 60 issues open as of writing (including those that are likely not achievable).

Finally, note that some of the items below were completed in the time between the progress demo and the submission of this document.

ID	L	Description	Status	Assigned
R01	1	DeadDrop agents shall issue heartbeat messages at a user-defined interval.	✓ 02/26	Lloyd
R02	1	DeadDrop shall allow the user to build new instances of agents, irrespective of the platform used by the server.	✓ 03/08	Lloyd
R03	1	DeadDrop shall allow the user to remotely execute shell commands through registered agents.	✓ 03/05	Lloyd
R05	1	DeadDrop shall allow users to remotely connect to the server interface.	✓ ~2023	Jann
R07	1	DeadDrop shall implement a covert communication protocol that communicates solely over a popular service.	✓ ~2023	Keaton
R08	1	DeadDrop shall implement a user authentication system.	✓	Jann/Brian

ID	L	Description	Status	Assigned
			03/08	
R09	1	DeadDrop shall provide a Svelte web interface, a reference implementation for interacting with all available backend endpoints.	✓ ~2023	Brian
R10	1	DeadDrop shall provide Pygin, a built-in reference implementation for an agent adhering to all available framework interfaces.	✓ ~2024	Lloyd
R16	1	Pygin shall execute commands concurrently.	✓ ~2024	Lloyd
R18	1	Pygin shall use supervisord to manage and configure its multi-process requirements.	✓ ~2023	Lloyd
R19	1	The DeadDrop server shall allow users to display and filter logs without requiring the logs to be exported.	✓ 12/01/23	Brian/Jann
R20	1	The DeadDrop server shall generate agent public/private key pairs in a cryptographically secure manner.	✓ 03/07	Lloyd
R22	2	DeadDrop agents shall provide a standard entrypoint script for installation that allows for the discovery of available commands and other metadata as a JSON document.	✓ 03/06	Lloyd
R29	2	The DeadDrop server shall implement a package manager allowing for the installation and availability of new agents and protocols without requiring assumptions of the server's directory structure.	✓ 02/29	Lloyd
R31	2	DeadDrop shall allow users to stand up fake websites to emulate popular services.	✓ ~2024	Keaton
R32	2	DeadDrop shall provide common statistics relating to messages sent and received over secure protocols.	✓ ~2024	Keaton/Brian
R33	3	Users shall be able to easily add new implementations of communication protocols.	✓ ~2024	Keaton/Lloyd
R44	3	DeadDrop shall handle asynchronous tasking with Celery.	✓ ~2024	Jann

III. Planned Requirements

Again, note that the following level convention is used:

- Level 1 requirements are those that we are confident will be done before Spring Break.
- Level 2 requirements are those that we are confident will be done before the final demo.
- Level 3 requirements are those that are nice (or necessary) in industry, but are more uncertain in terms of a timeline.

A circle denotes that work is ongoing *specifically* for that requirement; a blank status denotes that work has not yet started for that requirement.

ID	L	Description	Status	Assigned
R04	1	DeadDrop shall allow users to export logs to a standardized format.	◯	Jann/Brian
R06	1	DeadDrop shall encrypt messages using AES-CBC, with a minimum of a 128-bit key length.	◯	Lloyd
R11	1	DeadDrop shall record all actions initiated by a user or an agent in a standard format.	◯	Brian
R12	1	DeadDrop shall record all data transferred via communication protocols in a standard format.	◯	Jann
R13	1	DeadDrop shall use the Elliptic Curve Digital Signature Algorithm (ECDSA) to verify the authenticity and integrity of messages.	◯	Lloyd
R14	1	Pygin shall bundle itself as a PyInstaller-generated executable.		Lloyd
R15	1	Pygin shall contain portable, platform-specific executables for any dependencies needed.		Lloyd
R17	1	Pygin shall provide a command allowing users to execute arbitrary Python code.	◯	Lloyd
R21	1	The Svelte frontend shall use automated Lighthouse CI tests for accessibility, the results of which must be saved and tracked over time.		Brian
R23	2	DeadDrop shall allow the user to remove and remotely uninstall agents.		Jann
R24	2	DeadDrop shall implement a “guardrail” system, preventing the framework from acting on user-defined hosts.		Jann
R25	2	DeadDrop shall implement an “allowlist” system, forcing the		Jann

ID	L	Description	Status	Assigned
		framework to act only on user-defined hosts.		
R26	2	DeadDrop shall perform a user-defined action in the event an agent is unreachable for a specified number of attempts.		Keaton
R27	2	Pygin shall implement at least one example of a command renderer, whereby the response may be visualized by the frontend.		Brian
R28	2	Pygin shall provide the ability to transfer arbitrary files to and from the infected device.	◯	Keaton
R30	2	When presenting well-formatted responses from agents, DeadDrop may automatically visualize the response as a graph, chart, tree, or other visual listing.		Brian
R34	3	DeadDrop agents shall automatically discover accessible devices from the current device.		Jann
R35	3	DeadDrop agents shall be able to forward messages from other agents that do not have direct internet access.		Lloyd
R36	3	DeadDrop agents should not assume the availability of platform or distribution-specific binaries.		Keaton
R37	3	DeadDrop protocol implementations shall expose an easy method to visit the URLs being used as dead drop locations.		Keaton
R38	3	DeadDrop shall allow users to open a live terminal session with an agent using a standard communication protocol.		Keaton/Lloyd
R39	3	DeadDrop shall allow users to register “virtual” agents that represent discovered machines with no agent installed.	◯	Jann
R40	3	DeadDrop shall allow users to unregister virtual agents.	◯	Jann
R41	3	DeadDrop shall allow users to write reusable scripts to execute sets of shell or agent-native commands on a particular agent.		Jann
R42	3	The Svelte frontend shall provide users with desktop notifications of completed asynchronous tasks.		Jann/Brian
R43	3	Pygin shall implement a command that steals Discord session tokens, if installed, for demonstration purposes.	◯	Lloyd

IV. Time Worked

This includes **all** development since the fall prototype demo, as well as time spent in team meetings and discussing relevant design details.

Team Member	Hours	Contributions
Jann Arellano	105	- Asynchronous server tasking with Celery and Redis - Token-based authentication system with web interface - Implementation of RESTful API endpoints and functionality needed for web interface
Brian Buslon	80	- Token-based authentication system with web interface - Logic for rendering backend data to frontend - Development of frontend components and routes

Keaton Clark	90	- Team project website
		- Communication protocols:
		  - Generic video-based protocol (dddb) with extensive configurability for encoding
		  - Implementation of dddb over YouTube
		  - Partial implementation of dddb over Zoom
		  - Implementation of custom protocol over Craigslist

Lloyd Gonzales	120	- Pygin, the “reference” implementation of a DeadDrop agent - Package manager used to handle protocol and agent code bundles - Asynchronous, container-based payload generation system from installed packages - Schema generation used for agent configuration, validation, and dynamic form generation (for the web interface)