Security Orchestration Platform

Team: sdmay19-19
Advisor: Doug Jacobson

What is a Red Team/Adversary Simulation?

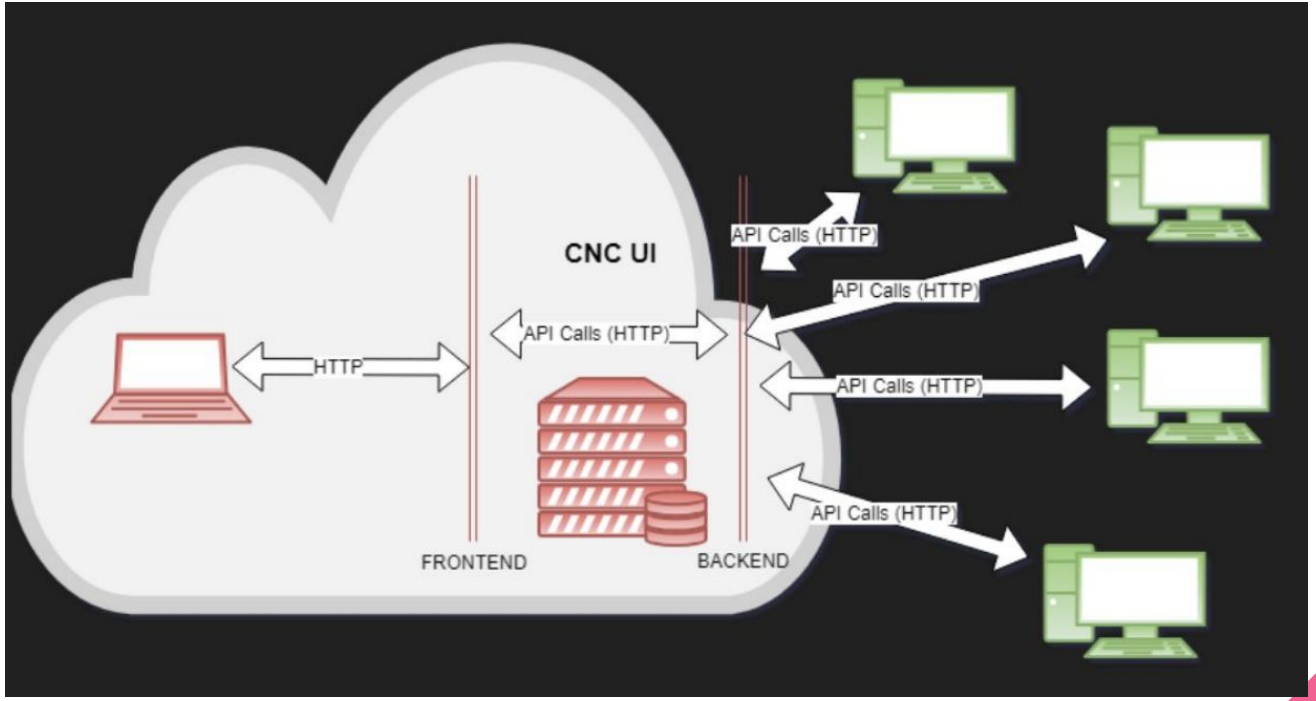
- Simulate an advanced attack against an organization
- Objective-based: “steal credit card numbers from PCI network”
- Blue team does not know about the red team assessment

https://render.fineartamerica.com/images/rendered/default/poster/8/10/break/images/artworkimages/medium/1/spy-vs-spy-mr-minor.jpg

Problem Statement

- Client faces issues with off the shelf tools being detected during red team engagements. Utilize a large number of manual processes which could be automated.
- Automation reduces the cost to deliver a red team assessment
- Custom tooling is extensible and is less likely to be detected by security solutions which are focused on detecting pre built tools

Conceptual Sketch



HW/SW/Technology Platform(s) used

- Semantic UI Frontend Framework
- Django Web Framework
- Django REST Framework (for APIs)
- Django Channels (for websockets)
- SQLite
- Docker
- C#
- Cuckoo

Functional Requirements

Bot

- Communicates with C2 via an encrypted REST API
- Tested and executable on recent versions of Windows
- Able to disconnect from a C2 and shutdown
- Supports domain fronting with Amazon Cloudfront and frontable domains
- Demonstrates persistence while remaining stealthy
- EDR solution bypass capabilities

C2

- Communicates over encrypted channel with multiple bots
- UI for sending commands to different bots
- Django backend and SemanticUI Frontend
- Multi-user creation/deletion/authentication
- Logs activity by users
- Realtime websockets for receiving data
- Building & configuring of malware
- App deployable with Docker
- Documentation/help for users

Non-Functional Requirements

Bot

- Able to destroy itself upon demand or if it cannot locate the C2
- Shall not be noticeable by the average user whose system is compromised
- Shall not have predictable network traffic (ie., beacon jitter)
- Configurable and supports multiple deployment options
- Secure against reverse engineering/losing source code/identifying the owner

C2

- Lengthy tasks shall be performed asynchronously
- User shall be able to navigate application freely without interrupting any ongoing processes
- Multiple users shall be able to access the application simultaneously
- Application shall not be accessible to general public
- Shall be quickly deployable in a temporary state

Project Milestones & Schedule

~ LEGEND ~
BOTH SUBTEAMS
BOT SUBTEAM
C2 SUBTEAM

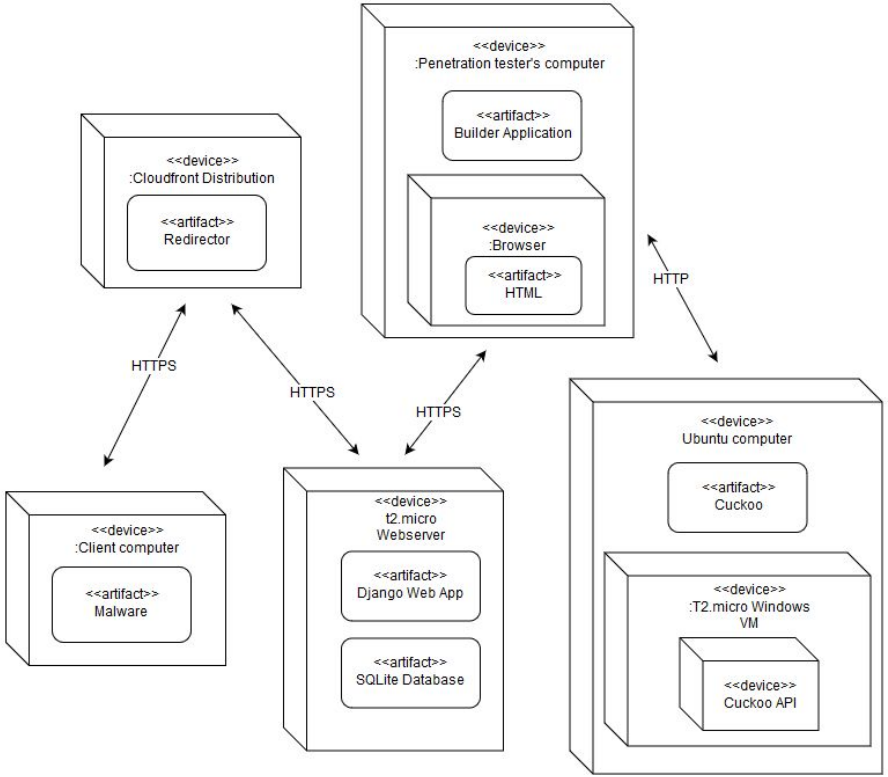
Task	WEEK 1 JAN 14	WEEK 2 JAN 21	WEEK 3 JAN 28	WEEK 4 FEB 4	WEEK 5 FEB 11	WEEK 6 FEB 18	WEEK 7 FEB 25	WEEK 8 MAR 4	WEEK 9 MAR 11	WEEK 10 MAR 25	WEEK 11 APR 1	WEEK 12 APR 8	WEEK 13 APR 15	WEEK 14 APR 22	WEEK 15 APR 29
Secure API channel															
Implement domain fronting															
Bypass EDR															
Expand bot persistence															
Implement bot destruction															
User authentication															
Detailed action logging															
Create malware builder (in-app)															
Enable websockets for realtime updates															
Dockerize the app															
Code testing															
Documentation															

Project Milestones & Schedule

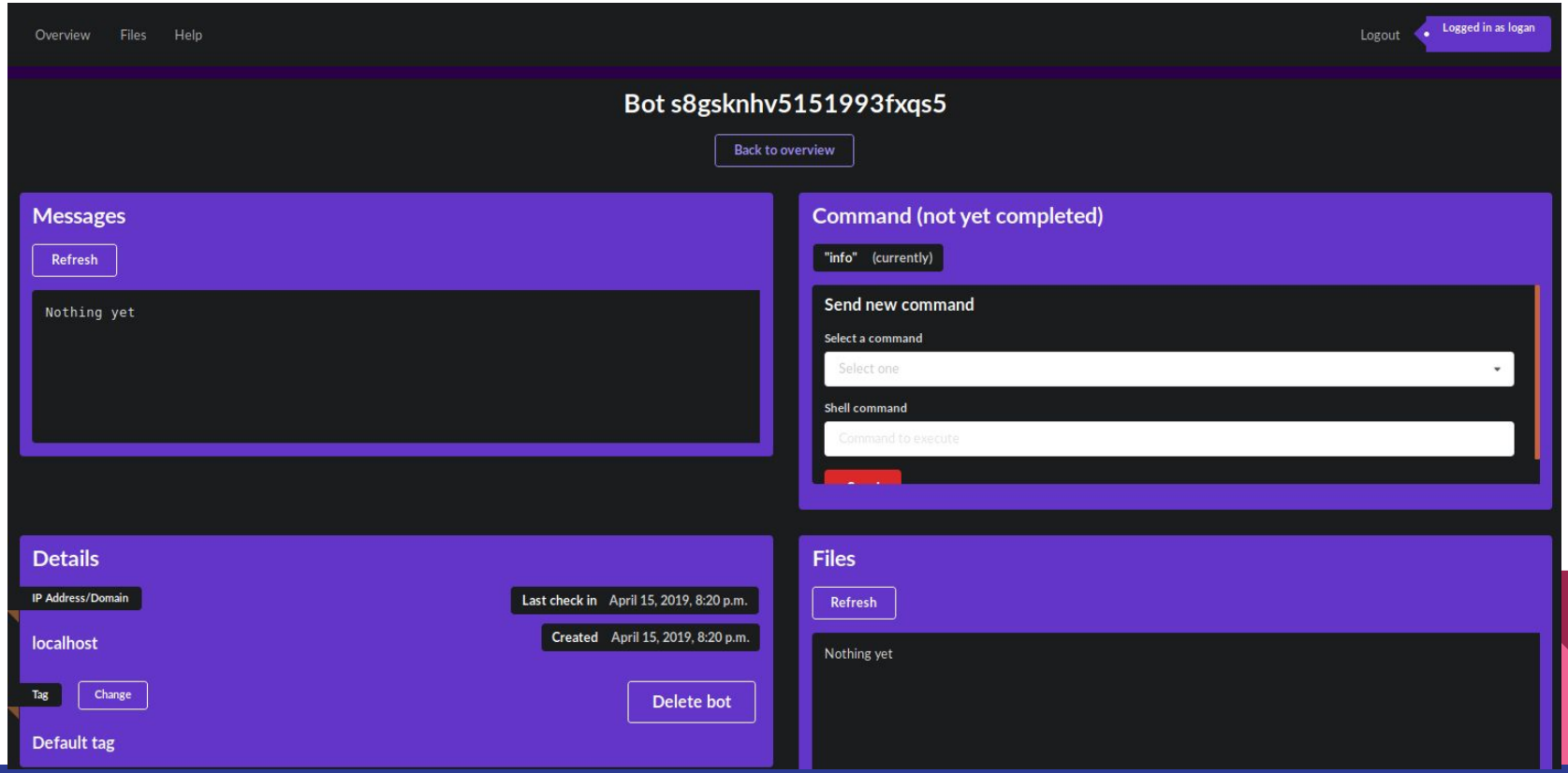
~ LEGEND ~
BOTH SUBTEAMS
BOT SUBTEAM
C2 SUBTEAM

Task	WEEK 1 JAN 14	WEEK 2 JAN 21	WEEK 3 JAN 28	WEEK 4 FEB 4	WEEK 5 FEB 11	WEEK 6 FEB 18	WEEK 7 FEB 25	WEEK 8 MAR 4	WEEK 9 MAR 11	WEEK 10 MAR 25	WEEK 11 APR 1	WEEK 12 APR 8	WEEK 13 APR 15	WEEK 14 APR 22	WEEK 15 APR 29
Secure API channel															
Implement domain fronting															
Bypass EDR															
Expand bot persistence															
Implement bot destruction															
User authentication															
Detailed action logging															
Create malware builder (in-app)															
Enable websockets for realtime updates															
Dockerize the app															
Code testing															
Documentation															

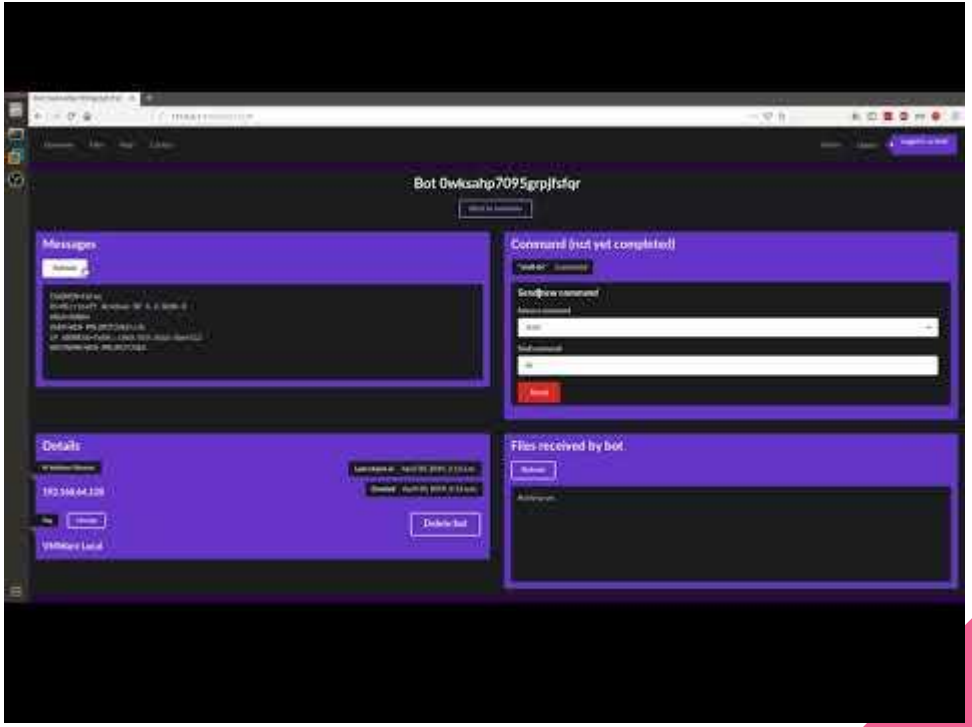
Functional Decomposition

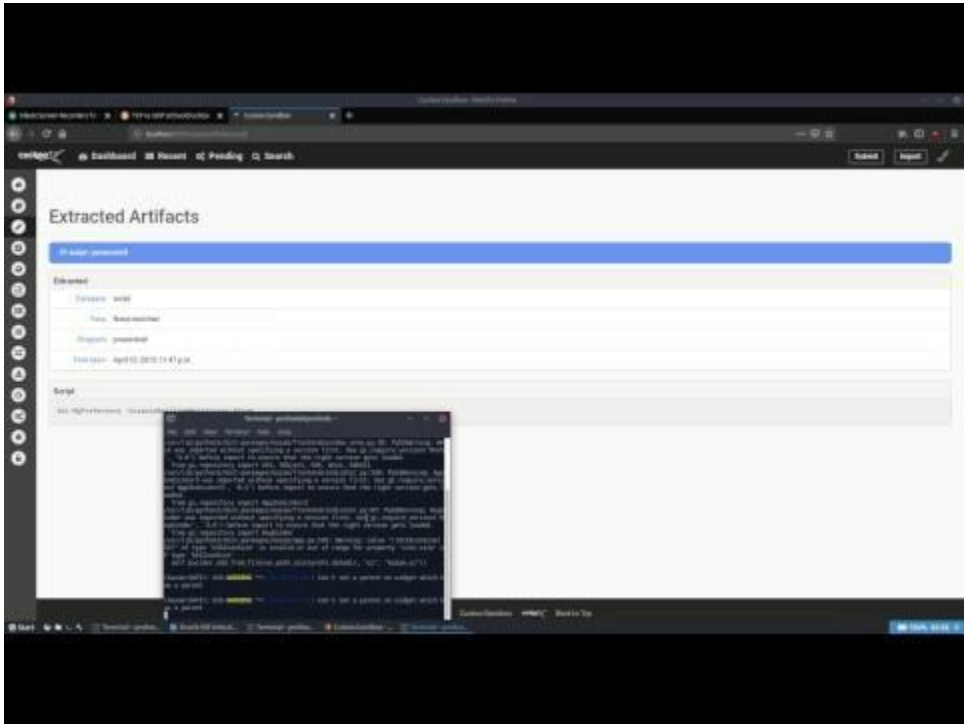


UI/UX Design Implementation



Custom Frontend Interface




Extracted Artifacts

Interesting Technique - Implant Builder

- Developed a new technique for building implants without recompiling the implant binary.
- Problem: Implant needs to be configured to connect to a given C2 server without being recompiled. Implant cannot rely on external configuration files, etc.
- Solution: Patch a compiled version of the implant at build time with configuration information.

Malware Builder

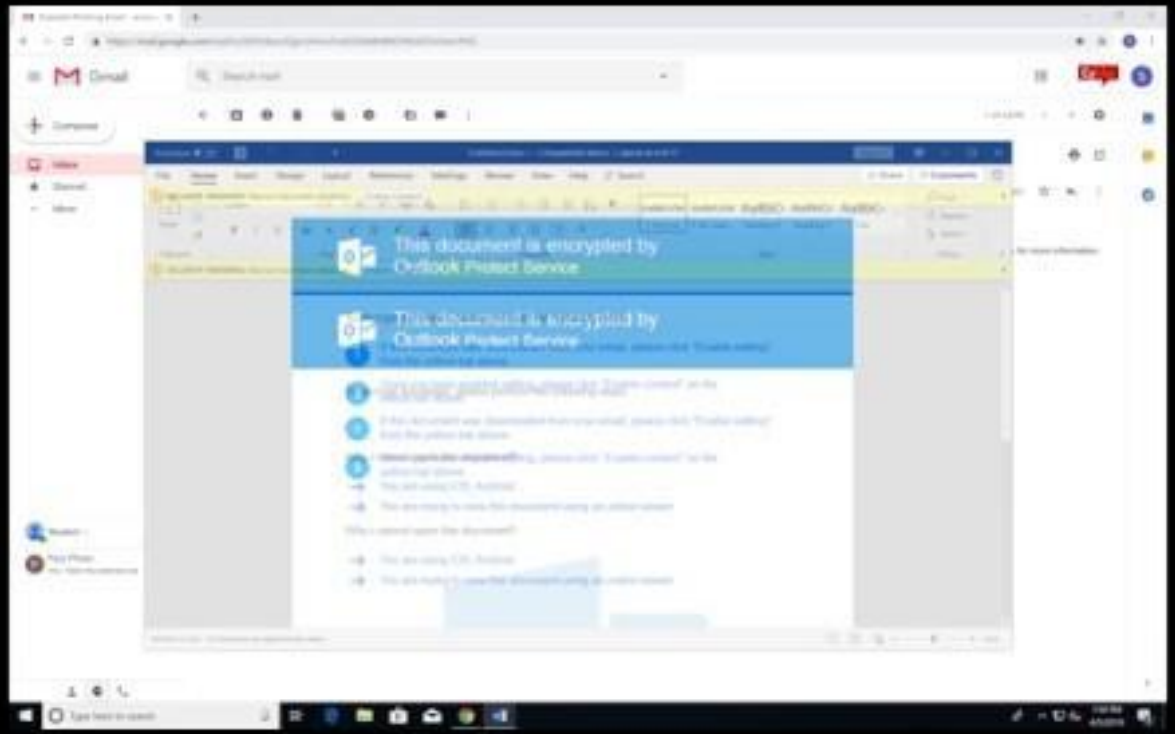
- User runs a separate application in order to patch the binary called the malware builder
- Uses Mono.Cecil to manipulate DotNet Intermediate Language (IL) code in order to modify implant configuration options.
- Inserts code into function call in order to return a specified value. In the template implant this returned value is blank.

```
27
28          public static string PWC_CLOUDFRONT_REDIRECTOR()
29          {
30              return "";
31          }
```


MediaFire
Gartner Report
Gartner Report
Compressed Archive (.RAR)
nextiva

Interesting Delivery Mechanism

- Developed mechanism for delivering malware using VBA macros in Microsoft Office products.
- Utilizes an egghunter to discover embedded OLE objects within the document to get around VBA size limitations.
- Displays decoy document to user when macros are enabled to avoid suspicions.


This document is encrypted by
Outlook Protect Service
This document is encrypted by
Outlook Protect Service

Test Plan

- A curated mixture of automated and manual testing
- Django/Python unit testing for C2
- Cuckoo Sandbox testing for Bot
- Manual integration testing for C2 and Bot
- User-level testing in Amazon's AWS Cloud Infrastructure with Client

```
(venv) dlimanow@gazelle:~/seniordesign/sdmay19-19/c2$ python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.......
----------------------------------------------------------------------
Ran 7 tests in 7.134s

OK
Destroying test database for alias 'default'...
(venv) dlimanow@gazelle:~/seniordesign/sdmay19-19/c2$
```

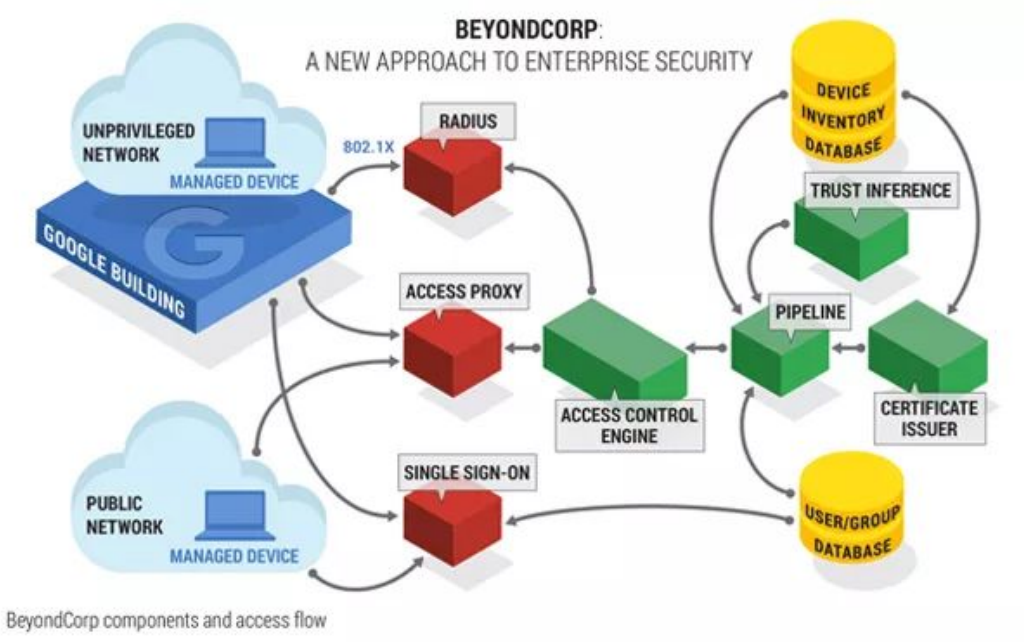
In order to ease the testing, the team made a decision to integrat
project. Cuckoo Sandbox is an open-source project that allows
deploying and integrating Cuckoo into the C2 server, a Red Te

Conclusion

- We demonstrated the product to the client (in-person) and completed all deliverables
- We learned a lot this semester and really enjoyed working with the client
- We hope the client or future senior design groups will be able to expand on our project

Future Work - Implement Google BeyondCorp For Red Team Infrastructure

- Consider implementing Zero Trust/Google BeyondCorp security model to protect access to client/customer information.
- Device Authentication (Certificates + TPM) + User Authentication (Password + FIDO U2F)
- Put implant server administrative interface behind an identity aware proxy (e.g. Cloudflare Access, Google IDP, etc.) or reverse proxy (e.g. Okta + SAML + FIDO U2F authentication)
- Can implement SSH authentication using time-limited certificates (e.g. https://github.com/Netflix/bless)



https://www.beyondcorp.com/img/no-vpn-security-3-full.jpg

Future Work - Malleable C2

- Expand implant for support for Malleable C2 mechanisms similar to that supported by Cobalt Strike
- Would allow for dynamically changing signature of network traffic sent by the implant.

```
http-get {

    set uri "/s/ref=nb_sb_noss_1/167-3294888-0262949/field-keywords=books";

    client {

        header "Accept" "*/*";
        header "Host" "www.amazon.com";

        metadata {
            base64;
            prepend "session-token=";
            prepend "skin=noskin;";
            append "csm-hit=s-24KU11BB82RZSYGJ3BDK|1419899012996";
            header "Cookie";
        }
    }
```

https://pbs.twimg.com/media/DKBL76RW4AA7K8y.jpg