Introduction

Problem and Importance:

2020 has seen a sharp rise in the frequency of cybersecurity attacks and with it an increase in their sophistication. These improvements make the attacks harder to spot, thus allowing them to compromise higher-value targets. Often these attacks will target people as the weakest link in the security chain and as a result phishing scams—a tactic which uses fraudulent emails to trick receivers into revealing sensitive information—are becoming ever more apparent. Once sensitive information—like a sysadmin password—has been stolen, the hacker may be able to steal integral data from the company, encrypt it, and then demand money for the encryption key. Since currently there is no way to decrypt the data without the key, the organization is forced to pay the ransom. As a result, prevention is the only method to combat these attacks. Preventing employees from accessing these fake domains can be difficult though even with proper safety training. The process of registering a fake domain, pushing malicious code and emailing victims can take less than an hour total. This speed paired with high volumes of attacks allows some fake domains to exist for weeks or months without being spotted and thus it is only a matter of time before someone makes a mistake and falls for the trap. Furthermore, these attacks are not only tricky to combat, but are also extremely costly—The FBI issued a PSA that estimates that over $12 billion were lost due to email phishing between 2013 and 2018.
Team Goals and Objectives:

Our main goal is to create an automated anti domain-spoofing service, which will assist our users in identifying, tracking and dismantling malicious websites. We plan to do so by creating a frontend web application which allows its users to configure groups and within them domains to monitor. A user can then configure at—a specified cadence—a dns record scan which will locate all potentially malicious domains which have similar domain names to those within the user’s group. Once the scan is complete our backend service will complete various automated checks on the phishing domains in order to rate their threat level and compose an analysis for each domain which can then be viewed on the web application. Furthermore, for high threat level domains a report will be emailed to a user’s configured addresses so the user (or their legal team) can take action against the creators of the site. As a final measure, our backend service will work with Novacoast’s firewall service to block navigation to these spoofing domains for all Novacoast firewall users.

Additionally, we all hope to gain various skills which will help us in our future careers. Foremost, we want to gain familiarity working with new pertinent technologies as these will not only be helpful in our future projects, but also provide good practice learning new concepts on the fly. We also hope to get experience working in remote environments and learn new methods of communication, which can help us in the future. Finally, we want to improve our ability to work together as a team and delineate tasks effectively and efficiently in order to maximize our speed to development.

Current Implementations:

To combat phishing, spam filter engines are typically the first line of defense. However, this solution cannot match the rampant speed at which attacks are being launched. Spam filter engines usually pull suspect domains from older data sources and therefore do not provide a real time solution. On the other hand, educational programs can provide real time solutions by bolstering security through teaching a company’s workforce to detect and avoid phishing scams. Although effective, companies are still vulnerable to phishing due to human error. Finally, there are products which allow users to locate potential spoofing domains, but these are not automated and thus require a user to search for their domain in order to get real time results.
Core Technical Advancements:

Our product seeks to leverage open source intelligence software—like dns twist—to handle the discovery of malicious domains for phishing prevention. We will then improve upon these by making recurrent calls to their APIs at a cadence specified by our users through our React Frontend. Thus, a large advancement in our product is the automation of subsequent checks for spoofing domains which—once configured for a user—will provide real time updates. Additionally, our product will contain a domain analysis tool on top of the automation, which assesses the risk that a domain is malicious using regex matching. It will then send automated reports to the corresponding company’s legal team or other specified points of contact with a breakdown of recent spoofing domains. Lastly, our software will expand its ability to prevent phishing attacks by integrating with Novacoast’s firewall to block users who accidentally travelled to malicious sites.

Assumptions:

- Users have their own domains to be monitored
- Users are subscribers of Novacoast’s services.
UI Mockups and Screenshots

Mockups:
Scan History

Scan Details
<table>
<thead>
<tr>
<th>Round</th>
<th>Domain</th>
<th>Created</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>amazon.com</td>
<td></td>
<td>35.18.98.39</td>
</tr>
<tr>
<td></td>
<td>amazon.com</td>
<td></td>
<td>20.05.60.233</td>
</tr>
<tr>
<td></td>
<td>amazon.com</td>
<td></td>
<td>200.9.33.90</td>
</tr>
<tr>
<td></td>
<td>amazon.com</td>
<td></td>
<td>124.068.77.32</td>
</tr>
<tr>
<td></td>
<td>amazon.com</td>
<td></td>
<td>125.15.222.48</td>
</tr>
<tr>
<td></td>
<td>amazon.com</td>
<td></td>
<td>248.149.192.11</td>
</tr>
<tr>
<td></td>
<td>amazon.com</td>
<td></td>
<td>257.36.202.132</td>
</tr>
<tr>
<td></td>
<td>amazon.com</td>
<td></td>
<td>128.302.65.36</td>
</tr>
<tr>
<td></td>
<td>amazon.com</td>
<td></td>
<td>242.32.26.25</td>
</tr>
<tr>
<td></td>
<td>amazon.com</td>
<td></td>
<td>335.302.122.179</td>
</tr>
<tr>
<td></td>
<td>amazon.com</td>
<td></td>
<td>87.122.45.52</td>
</tr>
</tbody>
</table>
System Architecture Overview

High Level Diagram:
Database Schema:

**Group**
- id INT
- name STRING
- emails TEXT
- start_at DATETIME
- recur STRING
- time_zone STRING
- created_at DATETIME
- updated_at DATETIME

**Domain**
- id INT
- name STRING
- group_id INT
- last_emailed DATETIME
- group OBJ
- active BOOL
- last_scanned DATETIME
- num_of_scans INT
- created_at DATETIME
- updated_at DATETIME

**Lookalike**
- id INT
- name STRING
- ip_address STRING
- domain_id INT
- domain OBJ
- found_on DATETIME
- updated_at DATETIME
- mx_record STRING
- nameservers STRING
- geop_country STRING
- creation_data DATETIME
**Sequence Diagrams**

**User Interactions:**

**User Updates Recurrence Information**

- **User**
  - Update Recurrence on Group Page

- **WebPage UI**
  - PUT /groups/<id> (Body = Recurrence)

- **Backend API Endpoint**
  - Query <id>
  - Return Group
  - Update Recurrence for Group

- **PostgreSQL DB**
  - HTTP Response

  Display new Recurrence
User views Scan Report Details

User

WebPage UI

Navigate to Group Page

GET /group/<id>

Query <id>

Return Group

HTTP Response

Display Group Information

Click View Recent Scan

GET /scan/<id>

Query <id>

Return Scan

HTTP Response

Display Scan Report Details Page

Backend API Endpoint

PostgreSQL DB
User views Scan Report Details

User
- Navigate to Login Page
- Display Login Page
- Enter Invalid Username and Password
  - POST /user (with credentials in body)
  - Query credentials
  - Return Unauthorized
  - HTTP Unauthorized Response
  - Display Invalid Credentials Error
- Enter Valid Username and Password
  - POST /user (with credentials in body)
  - Query
  - Return User Id
  - HTTP Response with User Id
- Log User In

WebPage UI

Backend API Endpoint

PostgreSQL DB
Class Interactions:

Scanning Domains for Lookalikes and Posting Results to the Database

User
Frontend Website
REST API
Database
Scheduler
RabbitMQ (message broker)
Worker

User inputs scan recurrence date and time
Sends data via HTTP POST
Create/Update recurrence date/time
Queries for recurrence date/time
Returns recurrence date/time
At specified date/time, sends a message specifying to start scanning domains
"Start scanning domains"
Scans domains and generates lookalikes

Sends lookalikes and scan results via HTTP POST
Create/Update lookalikes and scan results
Sending Emails of Scan Results to Group Admins

Worker → REST API → Database → SMTP API → Group

Queries for group emails via HTTP GET

Reads emails for group specified

Returns emails for group

Scan data and emails for group

Sends email(s) with scan data to group admins

Taking Screenshot of Lookalike and Storing Image in Amazon S3

Worker → Selenium API → REST API → S3 Bucket

Sends lookalike URL

Takes screenshot of webpage

Sends screenshot image via HTTP POST

Sends screenshot to be stored in S3
Prototyping Code, Tests, Metrics

GitHub: (frontend)  https://github.com/bredmond5/dns-twist-react
(backend)  https://github.com/bredmond5/dms-api-master

Sample Commits:

- Test case to check if query for lookalike can be filtered by domainId
  https://github.com/bredmond5/dms-api-master/commit/904577f705013b153087dff45b0a58004a686dff

- Test case to check if query for domain can be filtered by groupId
  https://github.com/bredmond5/dms-api-master/commit/904577f705013b153087dff45b0a58004a686dff

- Test case to check if query for lookalike can be filtered by scanId
  https://github.com/bredmond5/dms-api-master/commit/8419f4a7834b6b96e9ffe23c9c243299e2b4b5f2

- Test case to check routes for Scan model
  https://github.com/bredmond5/dms-api-master/commit/8419f4a7834b6b96e9ffe23c9c243299e2b4b5f2

- Worker now properly updates scan database after launching a dnsTwist
  https://github.com/bredmond5/dms-api-master/commit/90fe53cdadda190c260bbcce1666dad58654a60

- Frontend validates domain input
  https://github.com/bredmond5/dns-twist-react/commit/471ce5186ac3822e99ca2f69d01b545aae5e6f57

- Recurrence information added at group level
  https://github.com/bredmond5/dns-twist-react/commit/0010039a8b4a79b0598b660fb867cf7e5a3afc3

- Groups fetch domains from database
  https://github.com/bredmond5/dns-twist-react/commit/477d0325ca3822e99ca2f69d01b545aae5e6f57

- Sidebar fetches groups from database
  https://github.com/bredmond5/dns-twist-react/commit/3138410d9a6e15b415da20a205e5eff6de2cc0b4
#1: Receive Scheduled Reports

User Story: As a user I will receive reports on potential phishing domains with addresses similar to those I am registered for, so I can seek action to protect my company from phishing attacks.

Acceptance Criteria:
- Report emailed to addresses specified at correct frequency
- Report shows a table of each lookalike domain
- Email returns a table of similar domains for each domain managed by our group


Time Estimate: 10 person hours

#2: Create a Domain

User Story: As a user I can create a domain in the frontend so that all of its information is persisted in the database as well as the frontend.

Acceptance Criteria:
- Ability to create domains in the frontend, with emails, groups and recurrence information
- Persistence of data despite clearing of cache, refreshing, etc...

Card: [https://trello.com/c/w0OPGUHb/35-user-story-create-update-delete-a-domain](https://trello.com/c/w0OPGUHb/35-user-story-create-update-delete-a-domain)

Time Estimate: 10 person hours

#3: Create a Group

User Story: As a Tacklebox Admin I want to be able to create a group in the user interface, which stores a list of email addresses, domains, and recurrence information such that the data is saved in the database, and properly shows up in the UI.

Acceptance Criteria:
- I can add a group to the frontend, with emails/domains and recurrence information.
- I can refresh the page and the data persists.


Time Estimate: 10 person hours
#4: View Group Details

**User Story:** As a user I can navigate to a group and see its domains, with relevant lookalike counts from previous scan results.

**Acceptance Criteria:**
- Frontend renders count of lookalike domains within group
- Frontend renders most recent lookalike found
- Frontend renders the date of the most recent lookalike found

**Card:** [https://trello.com/c/NFtddJxy/37-user-story-view-group-details](https://trello.com/c/NFtddJxy/37-user-story-view-group-details)

**Time Estimate:** 12 person hours

#5: View Lookalike Details

**User Story:** As a user I can view details including date registered, IP Address, and similarity for each domain so that I can track malicious activity

**Acceptance Criteria:**
- Renders date registered in frontend
- Renders IP Address
- Renders similarity analysis to associated domain

**Card:** [https://trello.com/c/peXWoZVc/66-user-story-view-lookalike-details](https://trello.com/c/peXWoZVc/66-user-story-view-lookalike-details)

**Time Estimate:** 12 person hours

#6: View Scan Details

**User Story:** As a Tacklebox Admin I want to view a list of scans that took place on a group, click into a specific scan to see the exact lookalikes found, and click into a lookalike to drill into details.

**Card:** [https://trello.com/c/ii7GziIb/46-user-story-scan-details](https://trello.com/c/ii7GziIb/46-user-story-scan-details)

**Acceptance Criteria:**
- Renders time when scan started
- Renders time when scan ended
- Reports number of lookalikes found for that scan

**Time Estimate:** 12 person hours
#7: Continuous Front-end Deployment

**Use Case:** Whenever developers push code to the master branch of our frontend github repository, the updates are automatically deployed to the ECS service for the frontend.

**Card:** [https://trello.com/c/87oYRM8Y/40-use-case-continuous-front-end-deployment](https://trello.com/c/87oYRM8Y/40-use-case-continuous-front-end-deployment)

**Acceptance Criteria:**
- Continuously deploys front-end to AWS whenever the master front-end codebase updates
- Developers and individuals may access the continual deployed front-end via a URL

**Time Estimate:** 20 person hours

#8: Continuous Back-end Deployment

**Use Case:** Whenever developers push code to the master branch of our backend github repository, the updates are automatically deployed to the ECS service for the backend.

**Card:** [https://trello.com/c/WgSf1TMn/63-use-case-continuous-back-end-deployment](https://trello.com/c/WgSf1TMn/63-use-case-continuous-back-end-deployment)

**Acceptance Criteria:**
- Continuously deploys back-end to AWS whenever the master back-end codebase updates
- Developers can make requests to the continual deployed back-end via a URL

**Time Estimate:** 30 person hours

#9: Change recurrence information on a Group level

**User Story:** As a user I can update the recurrence information at the group level so that I can change the lookalike scan frequency by day, week, and month.

**Acceptance Criteria:**
- Ability to customize recurrence date, time zone and frequency
- Updating Recurrence information is reflected in both frontend and backend
- Lookalike scan updates frequency based off the new recurrence

**Card:** [https://trello.com/c/by02yZXX/67-user-story-change-recurrence-information-on-a-group-level](https://trello.com/c/by02yZXX/67-user-story-change-recurrence-information-on-a-group-level)

**Time Estimate:** 5 person hours

#10: Scheduled Database Update

**Use Case:** After a scheduled dns twist scan completes, it should update the database with the
new lookalikes found.

**Acceptance Criteria:**
- New lookalikes are present in the database after each scan
- The new lookalikes can be pulled from the database

**Card:** https://trello.com/c/mhB4T5Im/65-use-case-scheduled-database-update

**Time Estimate:** 5 person hours

**#11: Ping Legal Team**

**User Story:** As a user I want my legal team to be automatically contacted if there is a high enough similarity between my domain and a look alike so that they can reach out to the phishing domain directly and pursue legal action against them.

**Acceptance Criteria:**
- Report is sent to legal team if a certain similarity analysis is reached
- Report contains all necessary information for team to seek legal action

**Card:** https://trello.com/c/FYX6WBlt/64-user-story-ping-legal-team

**Time Estimate:** 5 person hours

**#12: View Screenshots of a Lookalike Domain Over Time**

**User Story:** As a user I want to view a screenshot of spoofed domains in order to access legal viability as well as document the state of each possible spoof domain.

**Card:**
https://trello.com/c/9VuDW2Th/68-user-story-view-screenshots-of-a-lookalike-domain-over-time

**Acceptance Criteria:**
- After a scan, screenshots of lookalikes are stored in the backend
- User can see screenshots over time that were gathered from the scanner

**Time Estimate:** 40 person hours

**#13: Return data back to open source intelligence**

**User Story:** As a Tacklebox Admin I want the information that is uncovered by my scans to be returned to open source intelligence so that clients of my website can be better protected.

**Acceptance Criteria:**
- Scans produce digestible data for open source intelligence systems
- Open source intelligence systems provide additional security to users

**Time Estimate:** 30 person hours

### #14: Domain name validation

**User Story:** As a Tacklebox Admin I want to add domains to a group and get real time validation so I can’t add invalid domains to my groups.

**Card:**

**Acceptance Criteria:**
- Valid domain is accepted as input
- Valid domain displays acceptance feedback to user
- Invalid domain is rejected as input
- Invalid domain displays rejection feedback to user

**Time Estimate:** 30 person hours

### #15: Authentication

**User Story:** As a Tacklebox Admin I want to be able to login so that I can only see the information relevant to my company.

**Acceptance Criteria:**
- A secure login with a email and password
- I can change my password using my email

**Card:** [https://trello.com/c/XmyQThhB/76-user-story-authentication](https://trello.com/c/XmyQThhB/76-user-story-authentication)

**Time Estimate:** 35 person hours

### #16: Delete a Domain

**User Story:** As a user I can remove a domain from its associated group so that I no longer track unnecessary domains.
Acceptance Criteria:

- Delete persists in the frontend despite clearing of cache, refreshing, etc...
- Deletion also removes entry from all rows in database where it occurs

Card: [https://trello.com/c/w0OPGUHb/35-user-story-create-update-delete-a-domain](https://trello.com/c/w0OPGUHb/35-user-story-create-update-delete-a-domain)

**Time Estimate**: 2 person hours

---

#17: Delete a Group

**User Story**: As a user I can remove a group and its associated domains so that I no longer track unnecessary groups.

**Acceptance Criteria**:

- Delete persists in the frontend despite clearing of cache, refreshing, etc...
- Deletion also removes entry from all rows in database where it occurs for both groups and domains connected to that group


**Time Estimate**: 2 person hours

---

#18: Block on Novacoast Firewall

**User Story**: As a person that runs Novacoast software, I want phishing domains to be blocked on my firewall if they are deemed to be malicious by Tacklebox.

**Card**: [https://trello.com/c/1L4Eylws/77-user-story-block-on-novacoast-firewall](https://trello.com/c/1L4Eylws/77-user-story-block-on-novacoast-firewall)

**Acceptance Criteria**:

- Malicious domains are automatically updated to a blocked URL list within Novacoast’s firewall
- User can no longer access sites marked malicious by Tacklebox

**Time Estimate**: 10 person hours

---

#19: Autoscaling worker containers

**User Story**: As a developer, I want to scale up and down worker containers to address the scheduled scans.

**Acceptance Criteria**:

- This uses less resources than constantly keeping the worker online
- This executes scans quicker since multiple scans could occur at the same time

Card: https://trello.com/c/gxjSQIJc/78-use-case-autoscaling-worker-containers

Time Estimate: 40 person hours

#20: Regex comparisons

User Story: As a Tacklebox Admin, I want similar domains that DNS Twist finds to be scanned for regex comparisons with my monitored website.

Acceptance Criteria:
- If the website is constructed in a way to look like the monitored website, it should be marked with a risk


Time Estimate: 50 person hours
Appendix

Technologies:

**Frontend** - React, Javascript, CSS

**Backend** - Flask (Python), SQLAlchemy

**Storage** - Amazon RDS, PostgreSQL, AWS S3

**Hosting** - AWS ECS, AWS ECR

**Internal Messaging** - RabbitMQ

**Domain Identification** - DNSTwist

**Containerization** - Docker