PhishAlarm URL Blocking

In early 2023, Fermilab cybersecurity team launched the PhishAlarm button for users to easily report suspicious phishing emails at the ease of a single click. When a user clicks the PhishAlarm button, the reported email is forwarded to the cybersecurity team for manual review.

Beholder is the main engine for Fermilab cybersecurity defenses. It integrates detectors, vulnerability scanners, and blocking mechanisms to protect the network against existing threats.

All links embedded in suspicious email reports are listed as separate entries in Beholder, under the section Unprocessed Phishing Reports URLs. A member of the cybersecurity operations team must review the contents of the email and decide if a malicious domain should be blocked centrally through Beholder or removed from the potential malicious domain list.

Alternatives to REST API

Two important methods were used with Selenium:

- pass_credentials_to_tap_dashboard: pass credentials from Beholder to the TAP dashboard.
- tap_event: perform the button click to trigger the TAP blocking event.

I took a new approach in Selenium, as Ruby controller actions and routes along with jQuery (JS library) .ajax() methods require REST API. Selenium is a viable alternative for button click events. I ultimately want tight integration with Beholder, so I directly used Ruby with the selenium-webdriver gem to locate the button element and simulate a button click programmatically.

Due to the code containing all URL and button data, and having credentials passed through Beholder, this provides a safe, transparent, and speedy alternative.

With the selenium-webdriver gem, a connection to TAP dashboard via the button element within the tap_event method will allow this to land in production, easing future blocking of URLs.

Automating Proofpoint Blocking

To prevent this blocked URL from appearing in future phishing emails to users, the cybersecurity operations team can apply an application-level block, so this URL is removed from emails sent to fnal.gov email addresses.

Illustration of the Proofpoint TAP Dashboard.

Investigating the Proofpoint REST API and endpoints for custom blocklist revealed that while there was implementation to get various pieces of information, there was no implementation to:

- GET for the URL blocklist entries
- SET (with authentication) for the blocking

I took a new approach in Selenium, as Ruby controller actions and routes along with jQuery (JS library) .ajax() methods require REST API.

Due to the code containing all URL and button data, and having credentials passed through Beholder, this provides a safe, transparent, and speedy alternative.

With the selenium-webdriver gem, a connection to TAP dashboard via the button element within the tap_event method will allow this to land in production, easing future blocking of URLs.

Investigating the Proofpoint REST API and endpoints for custom blocklist revealed that while there was implementation to get various pieces of information, there was no implementation to:

- GET for the URL blocklist entries
- SET (with authentication) for the blocking

I took a new approach in Selenium, as Ruby controller actions and routes along with jQuery (JS library) .ajax() methods require REST API.

Due to the code containing all URL and button data, and having credentials passed through Beholder, this provides a safe, transparent, and speedy alternative.

With the selenium-webdriver gem, a connection to TAP dashboard via the button element within the tap_event method will allow this to land in production, easing future blocking of URLs.