Evilginx 3.2 - Swimming With The Phishes

By Kuba Gretzky

Published: 2023-08-24 · Archived: 2026-04-29 02:09:22 UTC

Welcome back!

I've recently managed to find some free time to work on reverse proxy support for the latest Google updates and in the process I've made several additions to the Evilginx code base, which I think some of you will find useful.

To start, I wanted to give a big shoutout to Daniel (@dunderhay) for publishing a great post on how he used Evilginx to phish the Microsoft 365 ADFS environment and how he even made his modifications to succeed!

Evilginx is getting more love this year than in the last couple of years and I'm very happy about it. I have big plans for Evilginx, which I will announce soon, but first I wanted to give you a rundown of what the latest 3.2 update consists of.

I will start with the most significant changes.

Dynamic Redirection on Session Capture

One of the behaviours, that annoyed me when using Evilginx, was the fact that sometimes it was not possible to immediately redirect the phished user to the configured `redirect_url`, once all session tokens were captured. Evilginx could only redirect the browser once the targeted website attempted to navigate to a different page, on its own.

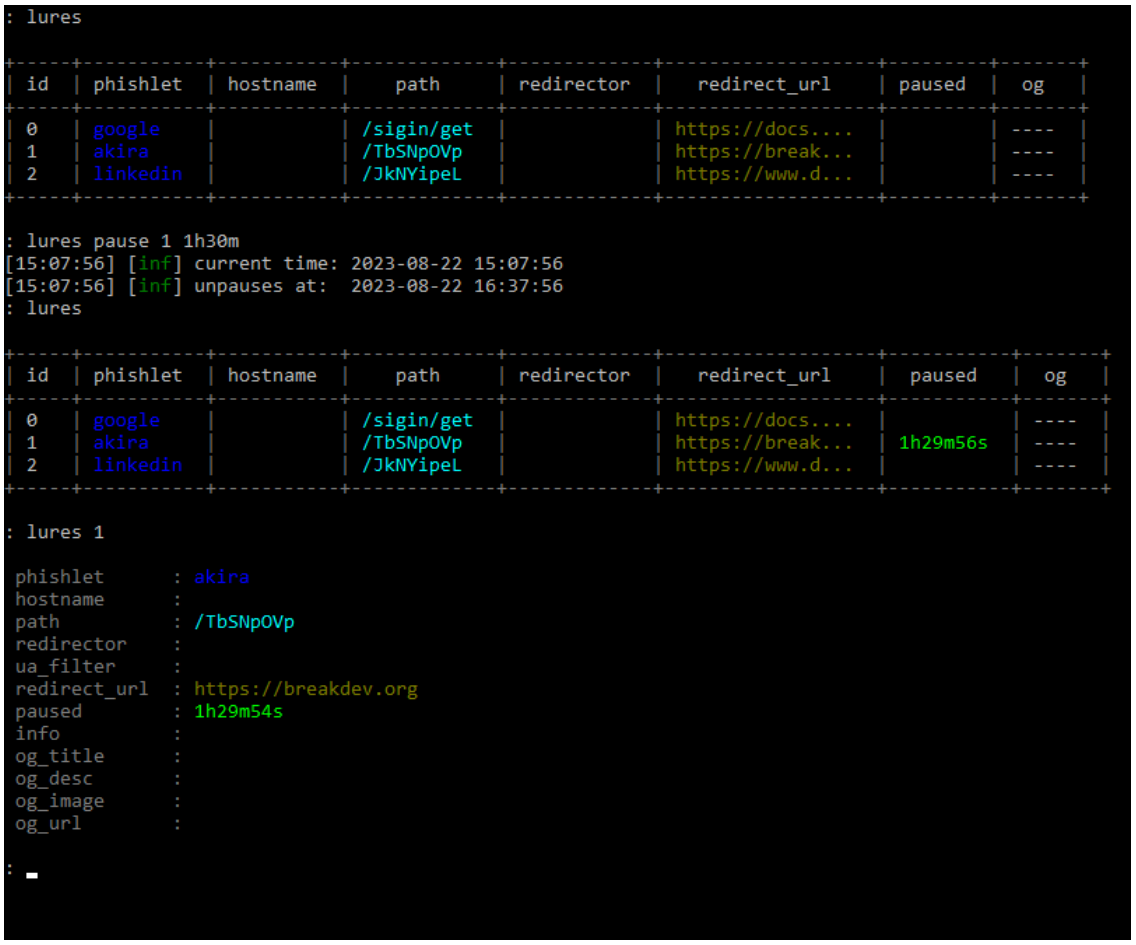
It made redirects not work on single-page applications. I learned it first-hand during the development of the Training Lab for my Evilginx Mastery course. The main page of the lab changes its contents dynamically and never navigates to a different URL. This means that once session tokens are captured by Evilginx, the tool is unable to redirect the user to `redirect_url` address.

In the 3.2 update, I've managed to solve the problem with injected JavaScript sending out HTTP long polling requests on every proxied page, to retrieve session capture status directly from the Evilginx proxy server in real-time. Evilginx will inject its own JavaScript code on every HTML page load, which will be responsible for querying `https://<phish_domain>/s/<phish_session_id>` infinitely. Evilginx proxy server will respond with a JSON structure, containing the `redirect_url` value only when the session is successfully captured. Otherwise, the connection will time out after 30 seconds and will be retried afterwards. Long polling allows Evilginx to let the injected script know that the session was captured immediately when it happens.

The script will then change the `window.location` URL to the retrieved `redirect_url` value, redirecting the user to a preconfigured page address. Redirection should now work great within Evilginx Mastery Training Lab.

Instead of HTTP long polling, I could've used WebSockets, but I wanted to keep it simple without the need to rely on external libraries, which would need to be injected as well.

Temporary Lure Pausing



Imagine a situation - you're on a phishing engagement and finally get to send out your phishing lures. Once the emails start arriving at the target inbox, the mailbox server opens them one by one and scans the HTML content of every phishing URL. The mail server then determines emails as phishing and they are sent to quarantine.

There are many ways to prevent automated scanners from seeing the content of your phishing pages, but the most straightforward method would be to simply hide your phishing pages for a brief moment, right before you send out the emails. Enough to hide their content from automated scanners, but not from the targeted user.

Now you can easily hide your lures from prying eyes by pausing them for a specific time duration with:

```
lures pause <id> <time_duration>
```

The best part is that you don't have to worry about unpausing a lure manually. Once the pause period expires, the lure with become active again and you will get a notification about it in the terminal. The pause state also persists between Evilginx restarts.

Interception of HTTP Requests

I found out that sometimes it would be useful to be able to block some of the proxied requests or have them return custom responses, without the proxied requests ever reaching the destination server.

Now you can detect specific requests within the new `intercept` section in your phishlets, which will match specific URL paths on domains within your `proxy_hosts` list. Once the request matches your filters, you will be able to detour the request and return your response with a custom HTTP status code.

```
intercept:
 - {domain: 'www.linkedin.com', path: '^\/report_error$', http_status: 200, body: '{"error":0}'', mime: "applic
 - {domain: 'app.linkedin.com', path: '^\/api\/v1\/log\/.*', http_status: 404}
```

In the example above, any request to `https://www.linkedin.com/report_error` will be intercepted and will return HTTP status `200` with response body `{"error":0}` and MIME type `application/json`.

The second entry will make sure that all requests to `https://app.linkedin.com/api/v1/log/<whatever>` will return `404 Not Found` HTTP response.

Redirect URL Added to Phishlets

Sometimes for the phishlet to work properly and to not interrupt the phished user's experience, it needs to redirect the user's browser right after session tokens are successfully captured. For now, the redirect would happen only if `redirect_url` was specified for the lure, used with the phishing engagement.

At times, it is important to have a default `redirect_url` specified, especially if we want the user to be redirected to the home page of the phished website, by design. Sometimes the redirection to the home page will happen automatically, but sometimes it needs to be enforced.

From this Evilginx version, you can set a default `redirect_url` in the phishlet you are creating to make sure the phished user is redirected, once session tokens are captured, even if `redirect_url` has not been set up for the given lure.

Unauthorized Request Redirects Per Phishlet

First of all, I've changed the name `redirect_url` from global config to `unauth_url`, to better illustrate its purpose and so it doesn't get confused with `redirect_url` set up in phishlets or lures.

IMPORTANT! Keep note that the URL you set for unauthorized request redirects may reset itself after the update, due to the name change.

Unauthorized URL or `unauth_url` holds the URL address where visitors will be redirected if they open any URL on the phishing domain, which doesn't correspond to any valid URL or if the lure is currently paused.

So far, it was possible to set up `unauth_url` globally, which would provide the same URL to redirect to for all active phishlets. With 3.2 you can now override the global `unauth_url` by specifying a value for each phishlet with:

```
phishlets unauth_url <phishlet> <url>
```

This feature was suggested by @0x_aalex who was also kind enough to submit a PR with his implementation. Thank you for that!

Tweaks and Fixes

Additionally to several new features, Evilginx has also received some QoL tweaks and fixes, which should improve the overall phishing performance.

Disabled caching of proxied content by web browsers

Sometimes it was especially frustrating to test the `sub_filters` of your phishlets because your web browser cached the content of your previous modification. Every time you made small changes and had to retest, it was required to clear the browser cache.

Starting from the 3.2 update, Evilginx will prevent web browsers from caching HTML, JavaScript and JSON content by injecting `Cache-Control: no-cache, no-store` HTTP header in proxied responses.

This should also make working with `js_inject` much more convenient.

JavaScript injected through external references

Normally when your phishlets inject JavaScript, through `js_inject` functionality, Evilginx would drop the whole script into the content of the HTML page within `<script>...</script>` tags. This approach was kind of messy, so I figured out a way to inject multiple scripts as external references, like this:

```
 <script type="application/javascript" src="/s/48d378a85f0867ef16bf0fd28deda0d4b30139c54805033803e7fdcbc31f293c/
<script type="application/javascript" src="/s/48d378a85f0867ef16bf0fd28deda0d4b30139c54805033803e7fdcbc31f293c.
```

Requests to download external JavaScript resources will be intercepted by Evilginx proxy and the response will be delivered from Evilginx directly, without ever being forwarded to the destination website.

This approach should make it possible to introduce dynamic JS obfuscation, in the future. (Stay tuned!)

Changelog

Here is the full changelog for Evilginx 3.2:

- Feature: URL redirects on successful token capture now work dynamically on every phishing page. Pages do not need to reload or redirect first for the redirects to happen.
- Feature: Lures can now be paused for a fixed time duration with `lures pause <id>`. Useful when you want to briefly redirect your lure URL when you know sandboxes will try to scan them.
- Feature: Added phishlet ability to intercept HTTP requests and return custom responses via a new `intercept` section.

- Feature: Added a new optional `redirect_url` value for phishlet config, which can hold a default redirect URL, to redirect to, once tokens are successfully captured. `redirect_url` set for the specific lure will override this value.
- Feature: You can now override globally set unauthorized redirect URL per phishlet with `phishlet unauth_url <phishlet> <url>`.
- Fixed: Disabled caching for HTML and Javascript content to make on-the-fly proxied content replacements and injections more reliable.
- Fixed: Blocked requests will now redirect using javascript, instead of HTTP location header.
- Fixed: Improved JS injection by adding `<script src"...">` references into HTML pages, instead of dumping the whole script there.
- Fixed: Changed `redirect_url` to `unauth_url` in global config to avoid confusion.
- Fixed: Fixed HTTP status code response for Javascript redirects.
- Fixed: Javascript redirects now happen on `text/html` pages with valid HTML content.
- Fixed: Removed `ua_filter` column from the lures list view. It is still viewable in lure detailed view.

Closing Thoughts

Hope you enjoy this update and there is more to come for Evilginx!

If you are interested in mastering the Evilginx phishing framework, consider checking out my Evilginx Mastery course:

In the upcoming weeks, I want to show off a sneak peek of Evilginx Pro and outline all of its extra features. Evilginx Pro will be a paid product I want to distribute only to vetted red teaming companies around the world.

I will post more details when I'm ready!

If you are interested in how to protect against reverse proxy phishing, do check out the talk I gave in May at x33fcon cybersecurity conference from this year:

For now, stay tuned and you can always follow me on Twitter @mrgretzky.

Source: https://breakdev.org/evilginx-3-2/