Evilginx 2.1 - The First Post-Release Update

By Kuba Gretzky

Published: 2018-09-10 · Archived: 2026-04-29 02:03:04 UTC

About 2 months ago, [I've released Evilginx 2]. Since then, a lot of you reported issues or wished for specific features.

Your requests have been heard! I've finally managed to find some time during the weekend to address the most pressing matters.

[>> Download Evilginx 2 from GitHub <<](https://github.com/kgretzky/evilginx2)

Here is what has changed and how you can use the freshly baked features.

Changelog - version 2.1

> **Developer mode added**

It is finally much easier to develop and test your phishlets locally. Start Evilginx with `-developer` command-line argument and it will switch itself into developer mode.
In this mode, instead of trying to obtain LetsEncrypt SSL/TLS certificates, it will automatically generate self-signed certificates.

Evilginx will generate a new root CA certificate when it runs for the first time. You can find the CA certificate at `$HOME/.evilginx/ca.crt` or `%USERPROFILE%\.evilginx\ca.crt`. Import this certificate into your certificate storage as a trusted root CA and your browsers will trust every certificate, generated by Evilginx.

Since this feature allows for local development, there is no need to register a domain at domain registrars. Just use any domain you want and set the server IP to `127.0.0.1` or your LAN IP:

```
 config domain anydomainyouwant.com
config ip 127.0.0.1
```

It is important that your computer redirects all connections to phishing sites, to your local IP address. In order to do that, you need to modify the `hosts` file.

First, generate the `hosts` redirect rules with Evilginx, for the phishlet you want:

```
 : phishlets hostname twitter twitter.anydomainyouwant.com
: phishlets get-hosts twitter

127.0.0.1 twitter.anydomainyouwant.com
```

```
127.0.0.1 abs.twitter.anydomainyouwant.com
127.0.0.1 api.twitter.anydomainyouwant.com
```

Copy the command output and paste it into your `hosts` file. The `hosts` file can be found at:
Linux: `/etc/hosts`
Windows: `%WINDIR%\System32\drivers\etc\hosts`

Remember to `enable` your phishlet and you can start using Evilginx locally (can be useful for demos too!).

> Authentication cookie detection was completely rewritten

There were some limitations in the initial implementation of session cookie detection, so I rewrote a significant portion of its code. Now, Evilginx is able to detect and properly use `httpOnly` and `hostOnly` flags, as well as `path` values for each captured cookie.

IMPORTANT! Previously captured sessions will not load properly with latest version of Evilginx, so make sure you backup your captured sessions before updating.

Evilginx will now properly handle cookie domains `.anydomain.com` vs `anydomain.com`. This is very important as I've noticed, during testing, the imported cookies will not provide a working session if cookie domain is set improperly.

The difference between each is, that cookie set for domain `.anydomain.com` will be sent in requests to `anydomain.com` and also to any sub-domain of `.anydomain.com` (e.g. `auth.anydomain.com`), while the cookie set for domain `anydomain.com` will be sent only with requests to `anydomain.com`.

I had to also update current phishlets to properly detect cookie domain names (with `.` prefix or without), so you may also need to update your private ones, accordingly.

> Regular expressions for cookie names and POST key names

It has come to my attention that some websites will dynamically generate cookie names or POST key names, based on user ID or some other factors. Since Evilginx was only able to look for fixed names, it would never be able to properly capture such cookies or intercept a username/password field in POST request.

Now, you can enter regular expressions for both cookie names and POST `user_regex` and `pass_regex`, by adding `regexp` to the string, after a comma `,` separator.

Here is the example. Let's say we want to capture a cookie that has the name `session_user_738299`, where `738299` is the user ID, which will be different for every user and thus will be a dynamic value. We can set up capturing of this cookie with regular expressions, like this (considering that the numerical value is always 6 digits):

```
auth_tokens:
 - domain: '.anydomain.com'
   keys: ['session_user_[0-9]{6},regexp']
```

This also can be done for POST keys. If we need to intercept a POST key, that will hold a username, with name `user[session_id_36273]`, we can do:

```
user_regex:
 key: 'user\[session_id_.*\],regexp'
 re: '(.*)'
```

Same applies to `pass_regex` of course.

> URL path detection for triggering session capture

You will find websites that set the session cookie ID before you even start typing your email in the login form. In such cases, Evilginx would detect the session cookie, capture it and since it was the last cookie it was meant to capture, it would consider the session captured. This would not be the case, since even no credentials were entered.

As smart people pointed out on Github, this can be remedied by detecting an HTTP request to specific URL path, which happens only after the user has successfully authenticated (e.g. `/home`).

Now you can add a new parameter in your phishlet `auth_urls`, where you provide an array of URL paths that will trigger the session capture. If we wanted to look for HTTP request to `/home`, we could do set it up like this:

```
auth_urls:
 - '/home'
```

With `auth_urls` set up in the phishlet, Evilginx will not trigger a session capture even when it considers all session cookies captured. These cookies will be stored only after the HTTP request to any of the specified URL paths happens.

IMPORTANT! URL paths in `auth_urls` are all considered regular expressions, so proper escaping may be required (also no need to add `,regexp` to each string)

There is now a cool trick that utilizes both `auth_urls` feature and regular expressions for cookie names.

If you ever come across a website that sends cookies in such a way that makes them impossible to detect with regular expressions, you can just opt for **capturing all the cookies** for a given domain and waiting for the URL trigger.

This is how you'd do it:

```
 auth_tokens:
  - domain: '.anydomain.com'
    keys: ['.*,regexp'] # captures all cookies for that domain
auth_urls:
  - '/home' # saves all captured cookies when this request is made
```

This is a very messy approach and I'd prefer to not see phishlets rely on that too much, but it can be done.

> Empty subdomains now work

There was a bug that would prevent phishlets to work for websites that do not use any subdomains for some of their hostnames. This is no longer an issue and as an example to prove that it is fixed, I've slightly modified the `twitter` phishlet to work with `twitter.com` hostname.

Wrapping up

Keep posting issues you are having with creating your own phishlets as I'm sure there will still be scenarios that will require some adjustments made to Evilginx.

You can update by pulling latest changes to the master branch. I will post a binary release once I confirm that everything is stable.

[>> Follow me on Twitter <<](https://twitter.com/mrgretzky)

[>> Download Evilginx 2 from GitHub <<](https://github.com/kgretzky/evilginx2)

I have some nice ideas still for upcoming releases like dynamic custom Javascript injection and forcing **"Remember Me"** check boxes by POST parameter injection.

Hope you are liking Evilginx so far.

Enjoy this update!
