Geekbang> 极客邦科技 InfoQ

QCon 全球软件开发大会 【北京站】2016

汽车、工控和物联网行业的0Day漏洞主动挖掘技术

邵强

Intemational Software Development Conference

内容概要

- 网络协议Fuzzing原理
- 基于Model-base千万级别fuzzing用例生成技术
- Safeguard增强安全功能检测技术
- 用fuzzing技术发现Heartbleed（“心脏出血”）漏洞
- 不同行业的fuzzing应用场景

（一）网络协议Fuzzing原理

- 随着时间的推移，软件承担了更多的责任

1980’ 1990’ 2000’ 2010’

✓

Quality
质量

✓ 🛡

Quality / Security
质量 / 网络安全

✓ 🛡 🔒

Quality / Security / Safety & Privacy
质量 / 网络安全 / 物理安全及隐私

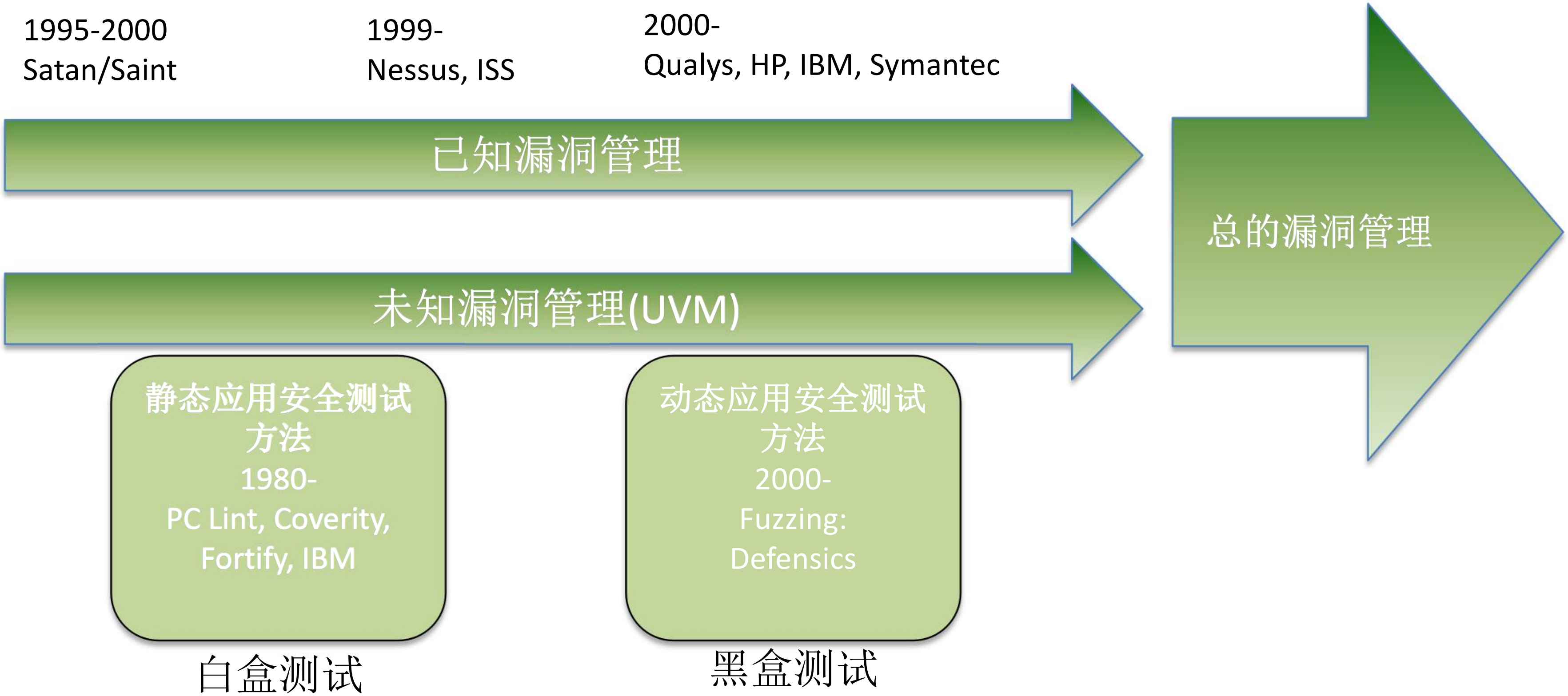
Financial Liability

Why Fuzzing???

“已知漏洞 vs 未知漏洞”

Known With Patch
Known With No Patch
Unknown
Vulnerabilities

已知和未知漏洞



1995-2000
Satan/Saint

1999-
Nessus, ISS

2000-
Qualys, HP, IBM, Symantec

已知漏洞管理

未知漏洞管理(UVM)

总的漏洞管理

静态应用安全测试方法
1980-
PC Lint, Coverity, Fortify, IBM

白盒测试

动态应用安全测试方法
2000-
Fuzzing:
Defensics

黑盒测试

底线: 所有的系统都具有漏洞.

- 两种互补的测试手段都需要被覆盖.

什么是 Fuzzing （模糊测试）

- Fuzzing （模糊测试）是向产品发送有组织的异常数据，试图引起系统错误或服务失败的过程。

- Fuzzing能暴露产品中可以被利用的漏洞

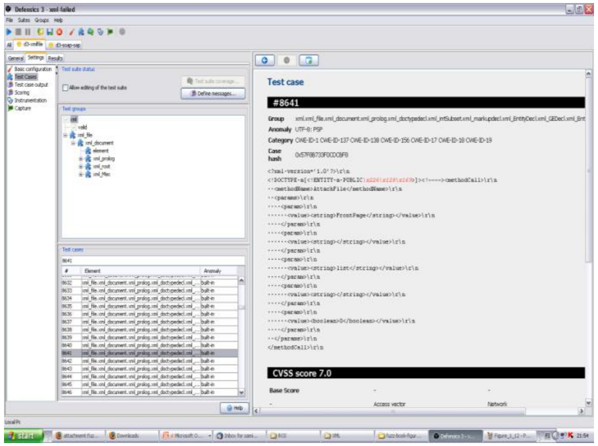
- 异常边界值
- 字段值溢进/溢出
- 格式化字符串
- IPv4/IPv6地址异常

……

Fuzzing（健壮性测试）的应用

- 基于黑盒的测试原理，无需接触源代码，通过报文交互进行攻击。
  - 任何有输入过程的软件都可以被Fuzzing：网络接口，设备驱动，用户界面….
- 黑客的杀手锏：用模糊 测试(fuzzing) 来寻找软件安全漏洞
  - 一旦发现漏洞，针对漏洞进行开发利用或者直接进行服务攻击
- 通常包含大量用例，通过高强度测试寻找可能存在的漏洞。
- 先敌而动，以敌人的方式来攻击自身，预先准备对策。
  - 原来越多的厂商开始采用Fuzzing技术测试产品的健壮性/安全性。

Fuzzing 测试过程



Test case

#8641

Group xml.xml_file.xml_document.xml_prolog.xml_doctypedecl.xml_intSubset.xml_markupdecl.xml_EntityDecl.xml_GEDecl.xml_Ent

Anomaly UTF-8: PSP

Category CWE-ID-1 CWE-ID-137 CWE-ID-138 CWE-ID-156 CWE-ID-17 CWE-ID-18 CWE-ID-19

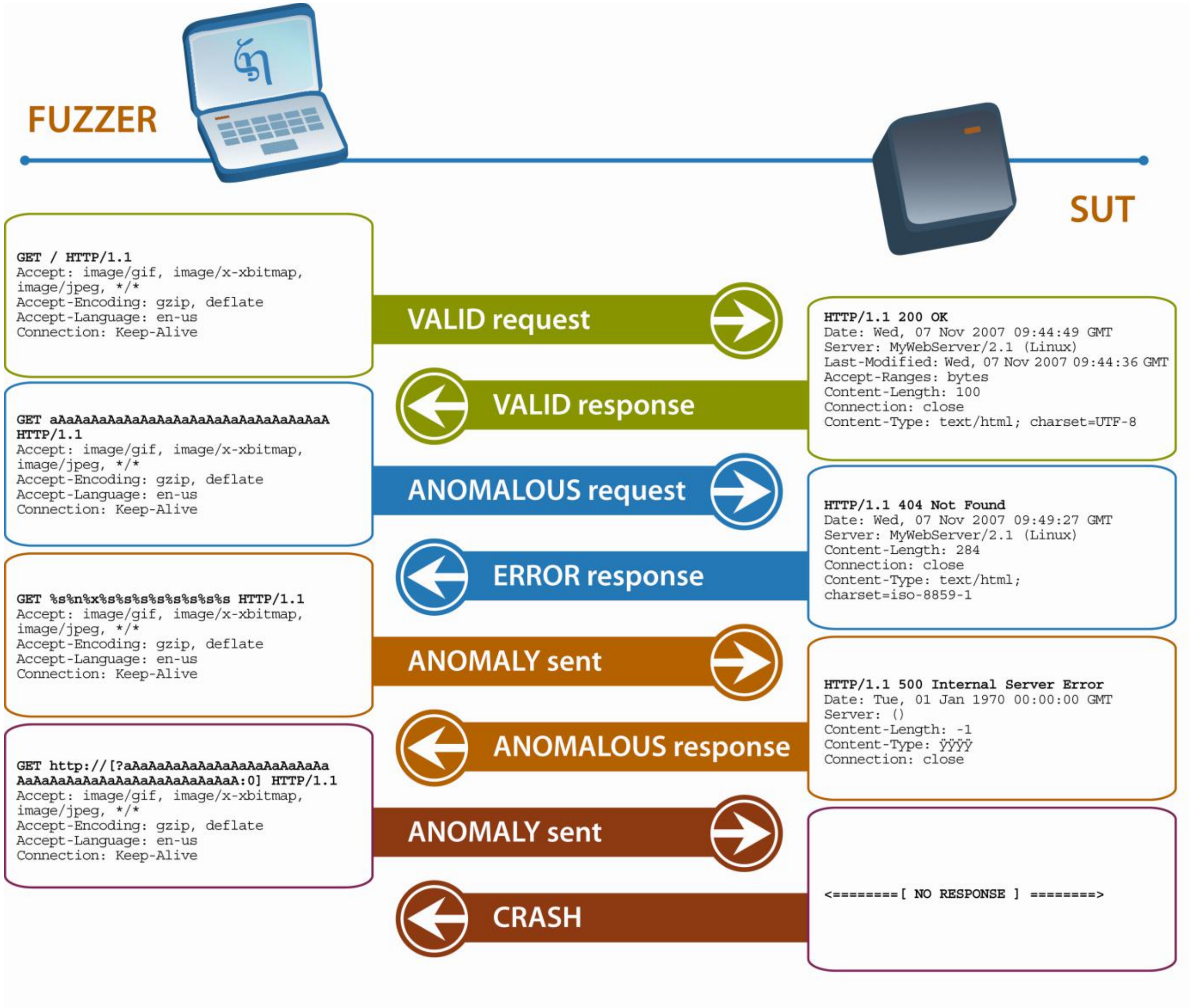
Case hash 0x57FBB733F0CDCBFB

```
<?xml·version='1.0'?>\r\n
<!DOCTYPE·a[<!ENTITY·a·PUBLIC'\x226\x128\x169>]><!----><methodCall>\r\n
··<methodName>AttachFile</methodName>\r\n
··<params>\r\n
····<param>\r\n
······<value><string>FrontPage</string></value>\r\n
····</param>\r\n
····<param>\r\n
······<value><string></string></value>\r\n
····</param>\r\n
····<param>\r\n
······<value><string>list</string></value>\r\n
····</param>\r\n
····<param>\r\n
······<value><string></string></value>\r\n
····</param>\r\n
····<param>\r\n
······<value><boolean>0</boolean></value>\r\n
····</param>\r\n
··</params>\r\n
</methodCall>\r\n
```

CVSS score 7.0

Base Score -

Access vector Network



```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap,
image/jpeg, */*
Accept-Encoding: gzip, deflate
Accept-Language: en-us
Connection: Keep-Alive
```

```
HTTP/1.1 200 OK
Date: Wed, 07 Nov 2007 09:44:49 GMT
Server: MyWebServer/2.1 (Linux)
Last-Modified: Wed, 07 Nov 2007 09:44:36 GMT
Accept-Ranges: bytes
Content-Length: 100
Connection: close
Content-Type: text/html; charset=UTF-8
```

```
GET aAaAaAaAaAaAaAaAaAaAaAaAaAaAaAaAaAaA
HTTP/1.1
Accept: image/gif, image/x-xbitmap,
image/jpeg, */*
Accept-Encoding: gzip, deflate
Accept-Language: en-us
Connection: Keep-Alive
```

```
HTTP/1.1 404 Not Found
Date: Wed, 07 Nov 2007 09:49:27 GMT
Server: MyWebServer/2.1 (Linux)
Content-Length: 284
Connection: close
Content-Type: text/html;
charset=iso-8859-1
```

```
GET %s%n%x%s%s%s%s%s%s%s%s HTTP/1.1
Accept: image/gif, image/x-xbitmap,
image/jpeg, */*
Accept-Encoding: gzip, deflate
Accept-Language: en-us
Connection: Keep-Alive
```

```
HTTP/1.1 500 Internal Server Error
Date: Tue, 01 Jan 1970 00:00:00 GMT
Server: ()
Content-Length: -1
Content-Type: ÿÿÿÿ
Connection: close
```

```
GET http://[?aAaAaAaAaAaAaAaAaAaAaAaAa
AaAaAaAaAaAaAaAaAaAaAaAaAaA:0] HTTP/1.1
Accept: image/gif, image/x-xbitmap,
image/jpeg, */*
Accept-Encoding: gzip, deflate
Accept-Language: en-us
Connection: Keep-Alive
```

```
<=========[ NO RESPONSE ] ========>
```

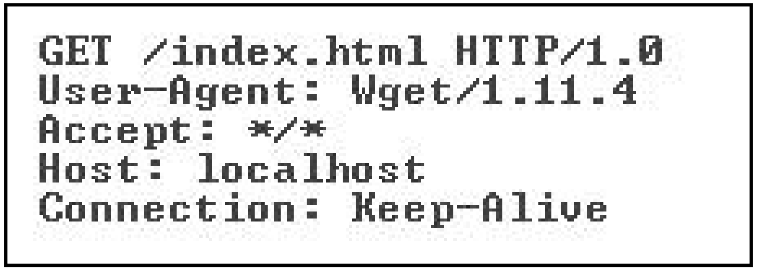
（二）基于Model-base千万级别fuzzing用例生成技术

- Model based 针对具体协议建模，对协议进行广泛深度覆盖
- 在协议建模基础上引入变异技术，生成海量测试用例
- 能够做到全协议覆盖和全状态支持
- 能够提供广泛的协议支持，涵盖通讯协议，文件格式，空口，工业控制，汽车电子，私有协议等范围。
- 便于构建自动化的测试流程，提高测试效率

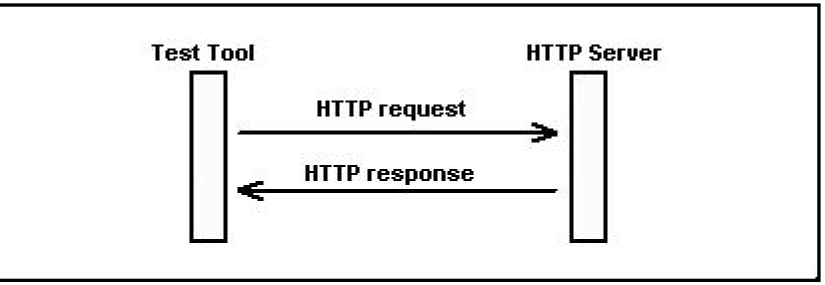
测试协议覆盖分类

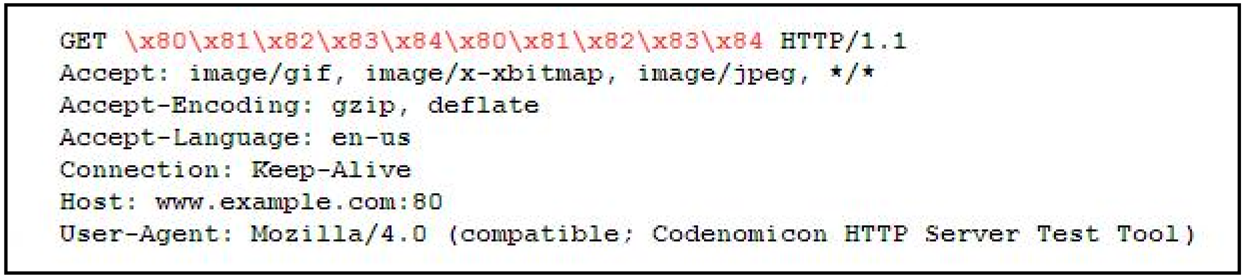
- Core Internet
- Net Management
- Routing
- Remote Access
- VPN
- VoIP
- 3G/4G/LTE
- Digital Media
- Email
- File System/Storage
- WLAN
- Link Management
- Bluetooth
- IPTV
- PDA/Smartphone
- Industrial Automation
- Archives
- Metro Ethernet
- General Fuzzer

HTTP 协议进行Fuzz 测试实例



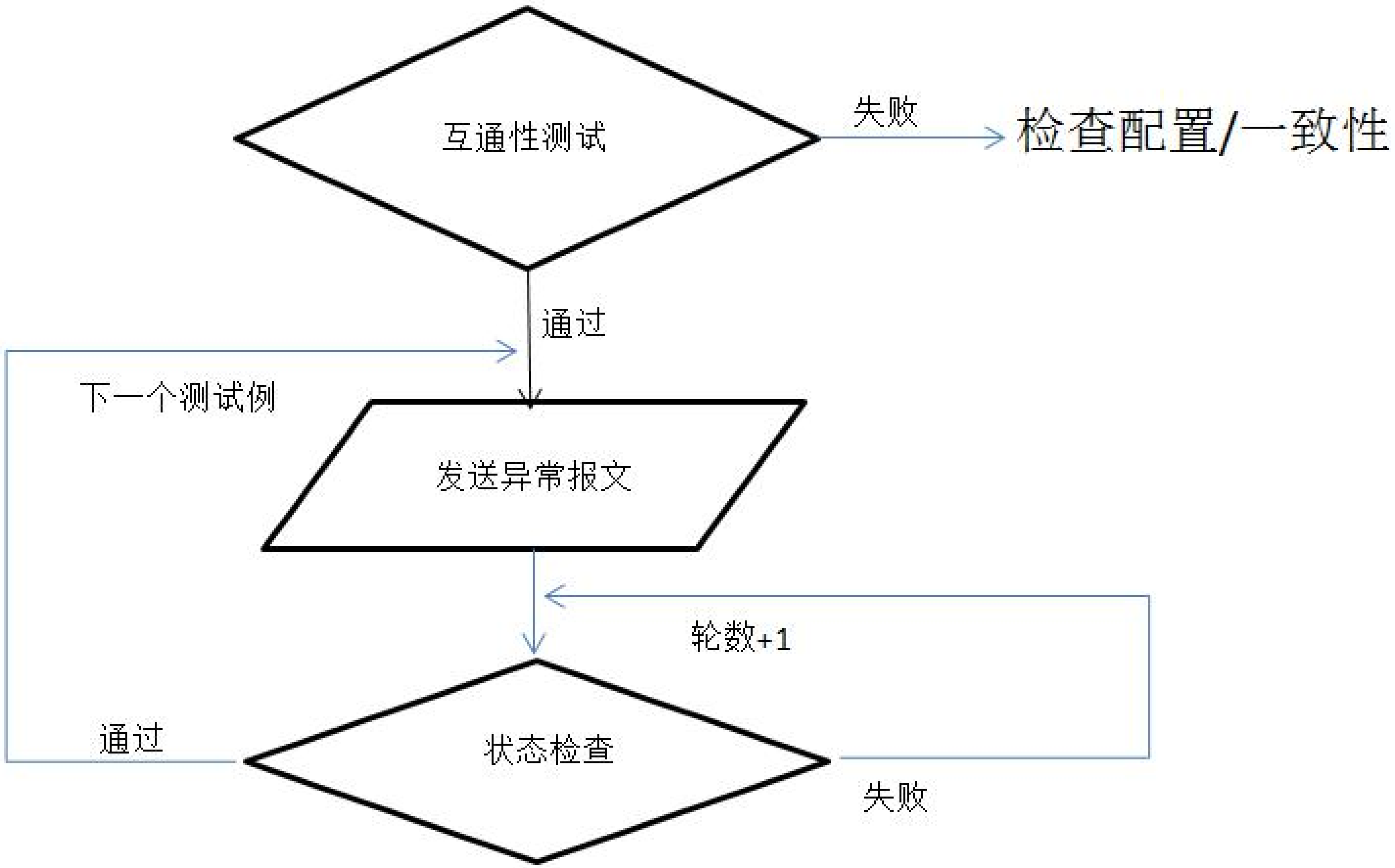
```
GET /index.html HTTP/1.0
User-Agent: Wget/1.11.4
Accept: */*
Host: localhost
Connection: Keep-Alive
```





```
GET \x80\x81\x82\x83\x84\x80\x81\x82\x83\x84 HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
Accept-Encoding: gzip, deflate
Accept-Language: en-us
Connection: Keep-Alive
Host: www.example.com:80
User-Agent: Mozilla/4.0 (compatible; Codenomicon HTTP Server Test Tool)
```

测试原理



（三）Safeguard增强安全功能检测技术

- Safeguard加强安全检测功能：在Fuzzing基础上，通过检查返回值并进行比对判断的相关安全检测方法。

- 放大攻击
- 旁路认证
- LDAP注入
- SQL注入
- 证书检查
- 随机度检查
- ……

（1）旁路认证

- 通过对正常的用户名密码/证书进行fuzzing，验证系统是否会通过验证。
- 验证通过，存在旁路验证风险。

（2）SQL注入

- 通过比较三种用例的返回结果，确定系统是否存在注入风险。
- 正常用例，永远为真用例和永远为假用例
- Valid case：

```
SELECT id FROM users WHERE name = '$USER' AND password ='$PASSWORD‘
```

- Always passing injection

```
$PASSWORD=1' OR 1=1
```

- Always failing injection

```
$PASSWORD=1' AND 1=2
```

（3）LDAP注入

- (&) ->Absolutely TRUE
  (|) ->Absolutely FALSE
- Valid case：

  (&(givenname=$first)(sn=$last))

- Always passing injection

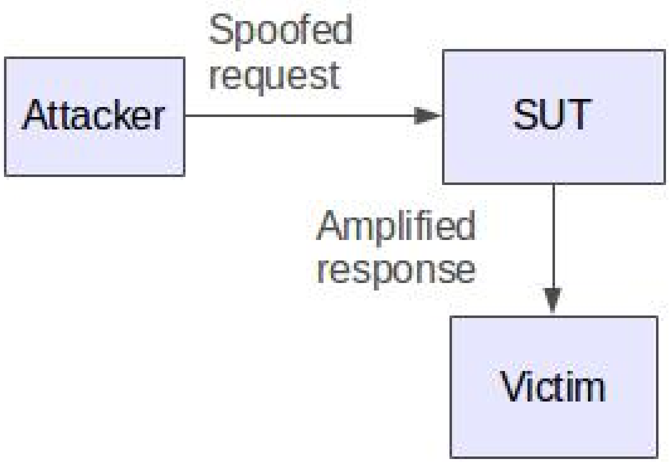
  “$last=doe)(&” => “(&(givenname=$first)(sn=doe)(&))”

- Always failing injection

  “$last=doe)(|” => “(&(givenname=$first)(sn=doe)(|))”

（4）放大攻击

- Server端到Client端的响应报文远大于Client端到Server端的请求报文。
- 常用于反射式DDoS攻击。

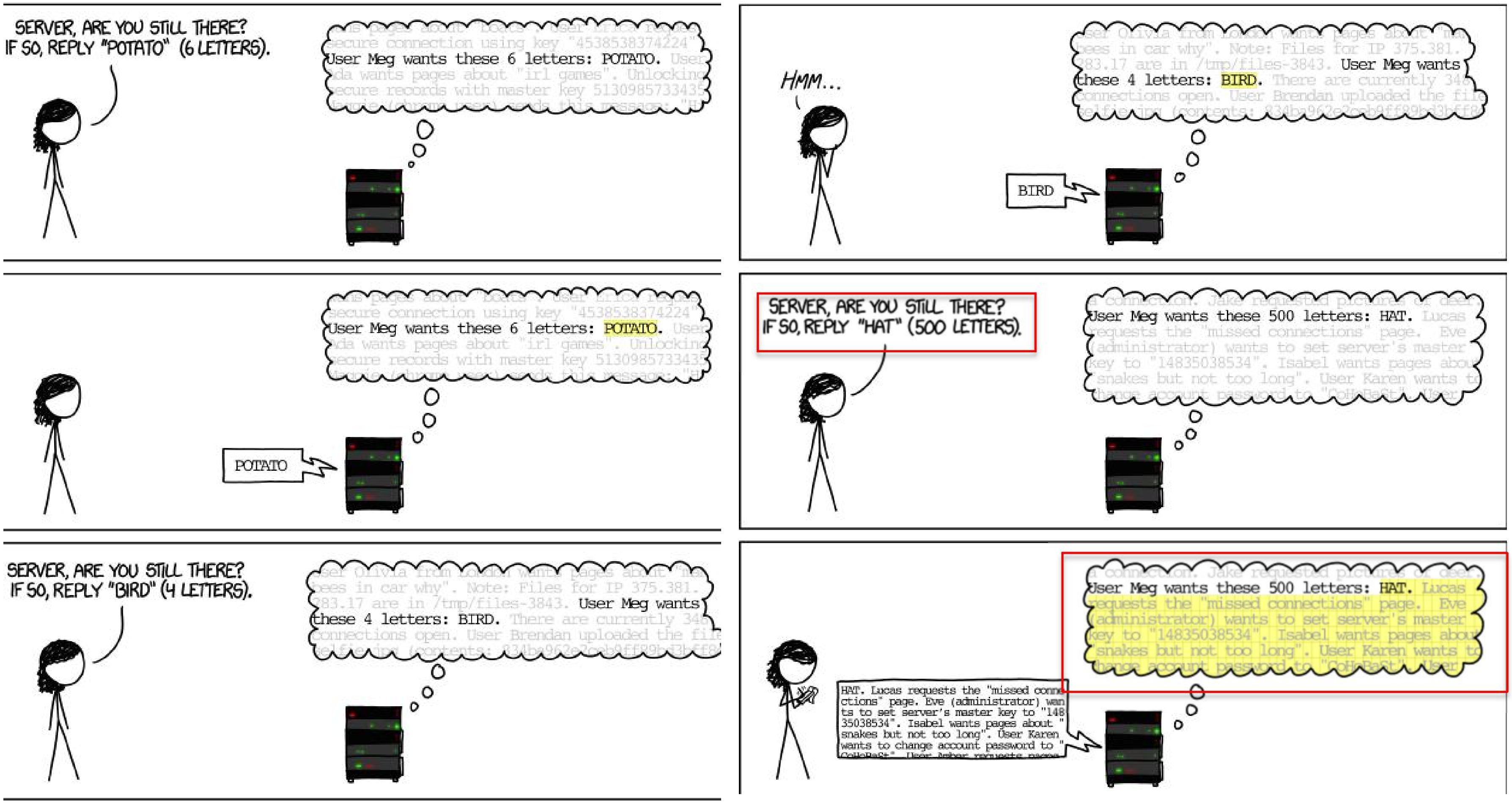


（四）用fuzzing技术发现Heartbleed（“心脏出血”）漏洞

- 2014年4月8日，“心脏出血”（Heartbleed）漏洞由安全厂商Codenomicon研发工程师和谷歌安全工程师分别独立发现，被誉为“互联网史上最为严重的漏洞”。

- OpenSSL在用于TLS/DTLS的Heartbeat扩展中，由于memcpy()没有在调用心跳请求包输入作为长度参数之前进行边界检查，导致攻击者可以以64KB/次的速度获取内存内容。

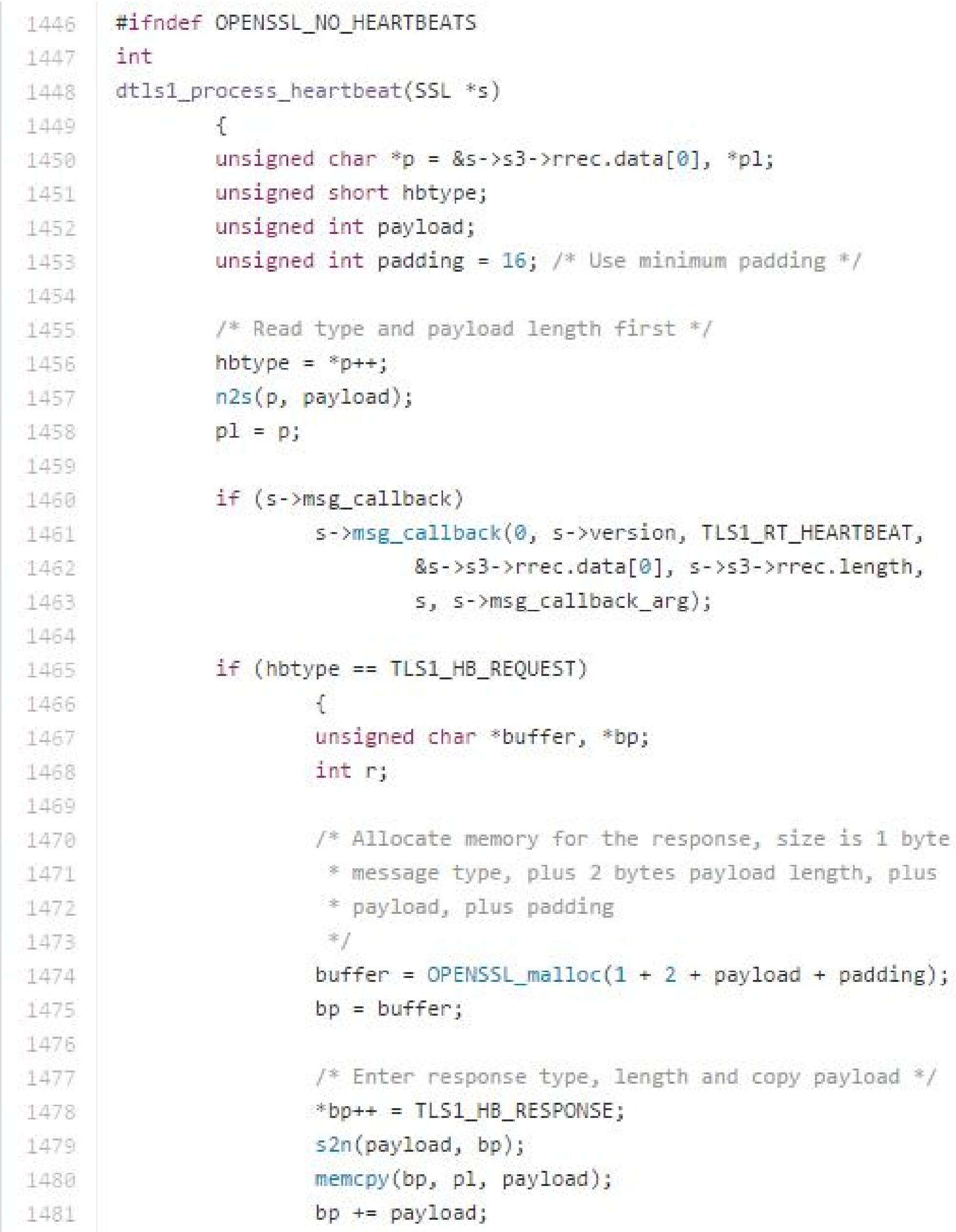
Heartbleed如何工作


SERVER, ARE YOU STILL THERE? IF SO, REPLY "POTATO" (6 LETTERS).
secure connection using key "4538538374224"
User Meg wants these 6 letters: POTATO. User
Ada wants pages about "irl games". Unlocking
secure records with master key 5130985733435
POTATO
secure connection using key "4538538374224"
User Meg wants these 6 letters: POTATO. User
Ada wants pages about "irl games". Unlocking
secure records with master key 5130985733435
SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).
User Olivia from London wants pages about "ma
bees in car why". Note: Files for IP 375.381.
283.17 are in /tmp/files-3843. User Meg wants
these 4 letters: BIRD. There are currently 346
connections open. User Brendan uploaded the file
selfie.jpg (contents: 834ba962e2ceb9ff89bd3bff8
HMM...
User Olivia from London wants pages about "ma
bees in car why". Note: Files for IP 375.381.
283.17 are in /tmp/files-3843. User Meg wants
these 4 letters: BIRD. There are currently 346
connections open. User Brendan uploaded the file
selfie.jpg (contents: 834ba962e2ceb9ff89bd3bff8
BIRD
SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
a connection. Jake requested pictures of deer.
User Meg wants these 500 letters: HAT. Lucas
requests the "missed connections" page. Eve
(administrator) wants to set server's master
key to "14835038534". Isabel wants pages about
snakes but not too long". User Karen wants to
change account password to "CoHoBaSt". User
a connection. Jake requested pictures of deer.
User Meg wants these 500 letters: HAT. Lucas
requests the "missed connections" page. Eve
(administrator) wants to set server's master
key to "14835038534". Isabel wants pages about
snakes but not too long". User Karen wants to
change account password to "CoHoBaSt". User
HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User Amber requests pages

http://xkcd.com/1354/

导致问题的代码

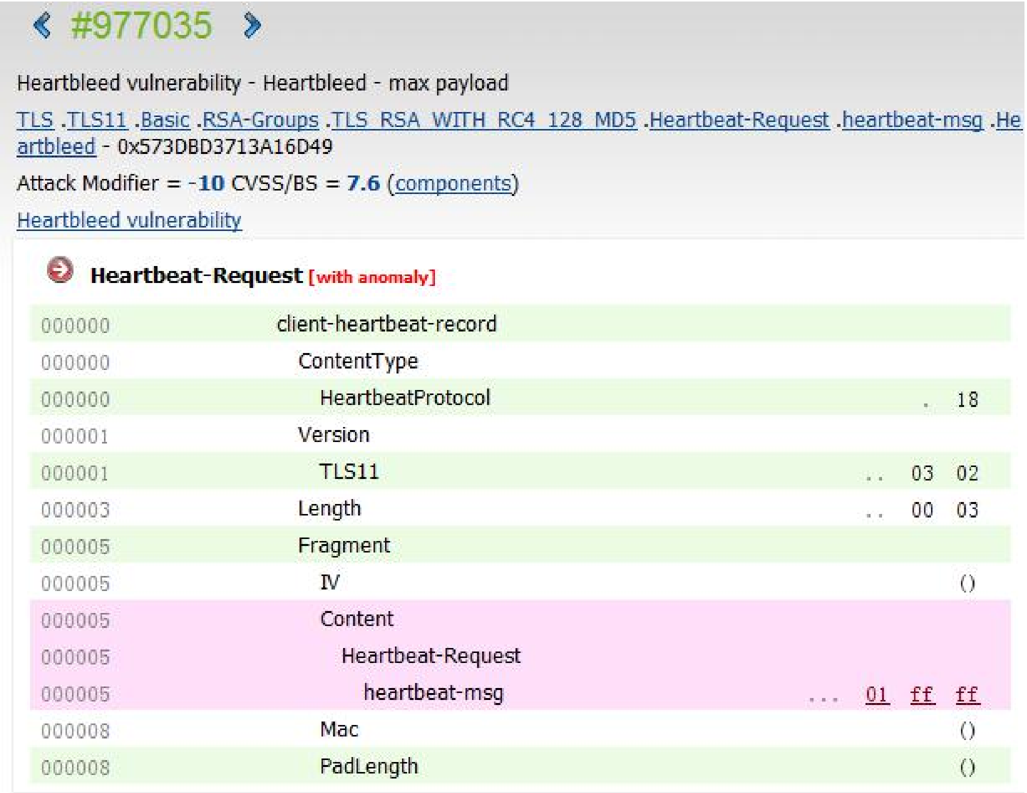
- Openssl-1.0.1c/ssl/d1_both.c



```c
#ifndef OPENSSL_NO_HEARTBEATS
int
dtls1_process_heartbeat(SSL *s)
	{
	unsigned char *p = &s->s3->rrec.data[0], *pl;
	unsigned short hbtype;
	unsigned int payload;
	unsigned int padding = 16; /* Use minimum padding */

	/* Read type and payload length first */
	hbtype = *p++;
	n2s(p, payload);
	pl = p;

	if (s->msg_callback)
		s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
			&s->s3->rrec.data[0], s->s3->rrec.length,
			s, s->msg_callback_arg);

	if (hbtype == TLS1_HB_REQUEST)
		{
		unsigned char *buffer, *bp;
		int r;

		/* Allocate memory for the response, size is 1 byte
		 * message type, plus 2 bytes payload length, plus
		 * payload, plus padding
		 */
		buffer = OPENSSL_malloc(1 + 2 + payload + padding);
		bp = buffer;

		/* Enter response type, length and copy payload */
		*bp++ = TLS1_HB_RESPONSE;
		s2n(payload, bp);
		memcpy(bp, pl, payload);
		bp += payload;
```

还原发现heartbleed的过程-1

通过修改heartbeat-request报文Payloadlength字段值，构造异常攻击报文，利用Heartbleed漏洞导致内存泄露。



#977035

Heartbleed vulnerability - Heartbleed - max payload

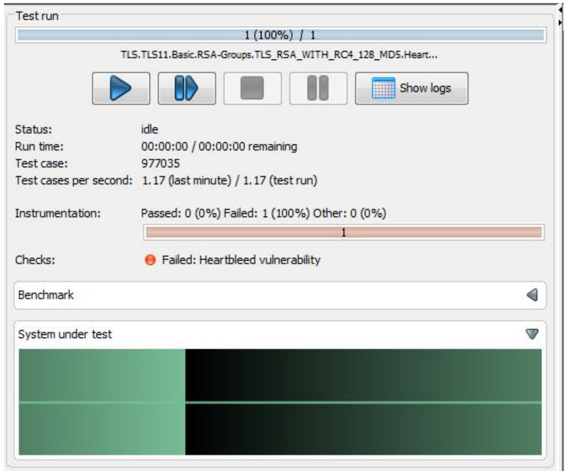
TLS .TLS11 .Basic .RSA-Groups .TLS_RSA_WITH_RC4_128_MD5 .Heartbeat-Request .heartbeat-msg .Heartbleed - 0x573DBD3713A16D49

Attack Modifier = -10 CVSS/BS = 7.6 (components)

Heartbleed vulnerability

Heartbeat-Request [with anomaly]

```
000000    client-heartbeat-record
000000      ContentType
000000        HeartbeatProtocol                          .  18
000001      Version
000001        TLS11                                 . .  03  02
000003      Length                                  . .  00  03
000005      Fragment
000005        IV                                             ()
000005        Content
000005          Heartbeat-Request
000005            heartbeat-msg                   . . .  01  ff  ff
000008        Mac                                            ()
000008        PadLength                                      ()
```



TLS.TLS11.Basic.RSA-Groups.TLS_RSA_WITH_RC4_128_MD5.Heart...

Status: idle
Run time: 00:00:00 / 00:00:00 remaining
Test case: 977035
Test cases per second: 1.17 (last minute) / 1.17 (test run)

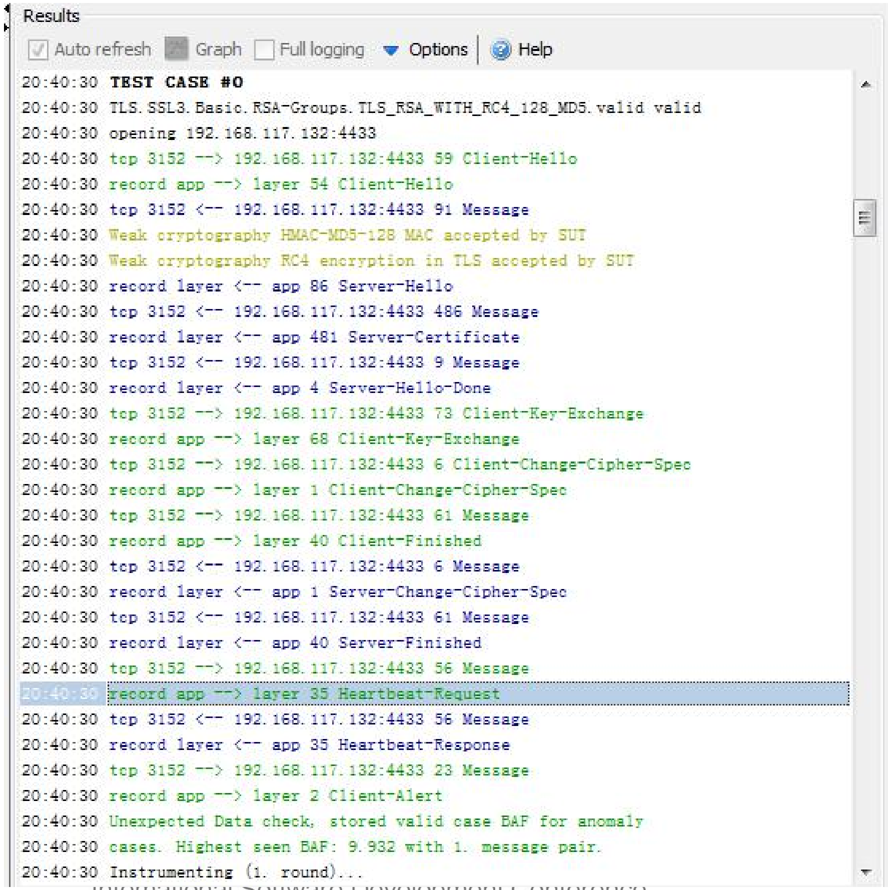
Instrumentation: Passed: 0 (0%) Failed: 1 (100%) Other: 0 (0%)

Checks: ● Failed: Heartbleed vulnerability

还原发现heartbleed的过程-2

在异常报文的攻击下，被测设备最多可返回64KB的内存数据。

（1）正常heartbeat报文

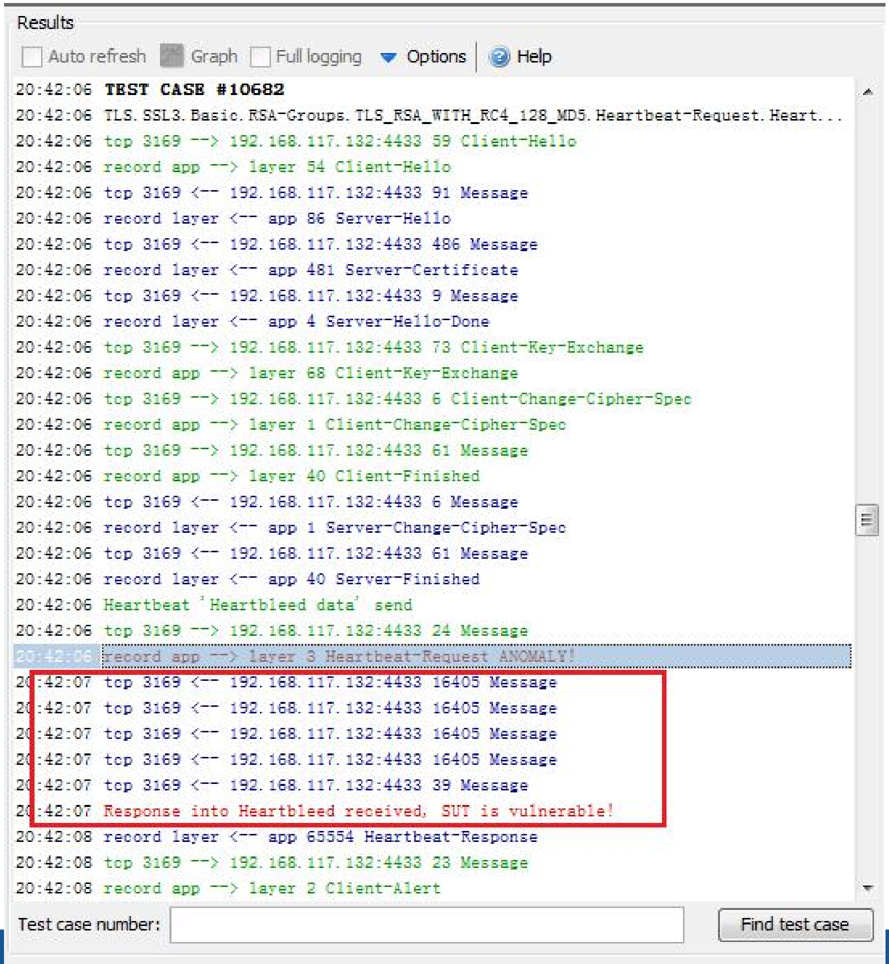


Results

☑ Auto refresh　Graph　□ Full logging　Options　Help

```
20:40:30 TEST CASE #0
20:40:30 TLS.SSL3.Basic.RSA-Groups.TLS_RSA_WITH_RC4_128_MD5.valid valid
20:40:30 opening 192.168.117.132:4433
20:40:30 tcp 3152 --> 192.168.117.132:4433 59 Client-Hello
20:40:30 record app --> layer 54 Client-Hello
20:40:30 tcp 3152 <-- 192.168.117.132:4433 91 Message
20:40:30 Weak cryptography HMAC-MD5-128 MAC accepted by SUT
20:40:30 Weak cryptography RC4 encryption in TLS accepted by SUT
20:40:30 record layer <-- app 86 Server-Hello
20:40:30 tcp 3152 <-- 192.168.117.132:4433 486 Message
20:40:30 record layer <-- app 481 Server-Certificate
20:40:30 tcp 3152 <-- 192.168.117.132:4433 9 Message
20:40:30 record layer <-- app 4 Server-Hello-Done
20:40:30 tcp 3152 --> 192.168.117.132:4433 73 Client-Key-Exchange
20:40:30 record app --> layer 68 Client-Key-Exchange
20:40:30 tcp 3152 --> 192.168.117.132:4433 6 Client-Change-Cipher-Spec
20:40:30 record app --> layer 1 Client-Change-Cipher-Spec
20:40:30 tcp 3152 --> 192.168.117.132:4433 61 Message
20:40:30 record app --> layer 40 Client-Finished
20:40:30 tcp 3152 <-- 192.168.117.132:4433 6 Message
20:40:30 record layer <-- app 1 Server-Change-Cipher-Spec
20:40:30 tcp 3152 <-- 192.168.117.132:4433 61 Message
20:40:30 record layer <-- app 40 Server-Finished
20:40:30 tcp 3152 --> 192.168.117.132:4433 56 Message
20:40:30 record app --> layer 35 Heartbeat-Request
20:40:30 tcp 3152 <-- 192.168.117.132:4433 56 Message
20:40:30 record layer <-- app 35 Heartbeat-Response
20:40:30 tcp 3152 --> 192.168.117.132:4433 23 Message
20:40:30 record app --> layer 2 Client-Alert
20:40:30 Unexpected Data check, stored valid case BAF for anomaly
20:40:30 cases. Highest seen BAF: 9.932 with 1. message pair.
20:40:30 Instrumenting (1. round)...
```

（2）畸形heartbeat报文



Results

□ Auto refresh　Graph　□ Full logging　Options　Help

```
20:42:06 TEST CASE #10682
20:42:06 TLS.SSL3.Basic.RSA-Groups.TLS_RSA_WITH_RC4_128_MD5.Heartbeat-Request.Heart...
20:42:06 tcp 3169 --> 192.168.117.132:4433 59 Client-Hello
20:42:06 record app --> layer 54 Client-Hello
20:42:06 tcp 3169 <-- 192.168.117.132:4433 91 Message
20:42:06 record layer <-- app 86 Server-Hello
20:42:06 tcp 3169 <-- 192.168.117.132:4433 486 Message
20:42:06 record layer <-- app 481 Server-Certificate
20:42:06 tcp 3169 <-- 192.168.117.132:4433 9 Message
20:42:06 record layer <-- app 4 Server-Hello-Done
20:42:06 tcp 3169 --> 192.168.117.132:4433 73 Client-Key-Exchange
20:42:06 record app --> layer 68 Client-Key-Exchange
20:42:06 tcp 3169 --> 192.168.117.132:4433 6 Client-Change-Cipher-Spec
20:42:06 record app --> layer 1 Client-Change-Cipher-Spec
20:42:06 tcp 3169 --> 192.168.117.132:4433 61 Message
20:42:06 record app --> layer 40 Client-Finished
20:42:06 tcp 3169 <-- 192.168.117.132:4433 6 Message
20:42:06 record layer <-- app 1 Server-Change-Cipher-Spec
20:42:06 tcp 3169 <-- 192.168.117.132:4433 61 Message
20:42:06 record layer <-- app 40 Server-Finished
20:42:06 Heartbeat 'Heartbleed data' send
20:42:06 tcp 3169 --> 192.168.117.132:4433 24 Message
20:42:06 record app --> layer 3 Heartbeat-Request ANOMALY!
20:42:07 tcp 3169 <-- 192.168.117.132:4433 16405 Message
20:42:07 tcp 3169 <-- 192.168.117.132:4433 16405 Message
20:42:07 tcp 3169 <-- 192.168.117.132:4433 16405 Message
20:42:07 tcp 3169 <-- 192.168.117.132:4433 16405 Message
20:42:07 tcp 3169 <-- 192.168.117.132:4433 39 Message
20:42:07 Response into Heartbleed received, SUT is vulnerable!
20:42:08 record layer <-- app 65554 Heartbeat-Response
20:42:08 tcp 3169 --> 192.168.117.132:4433 23 Message
20:42:08 record app --> layer 2 Client-Alert
```

Test case number: 　Find test case

（五）不同行业的Fuzzing应用场景

Fuzzing不仅适用于IT行业

- 汽车电子：Canbus
- 工业控制系统：Modbus、CIP、Profinet、IEC62443标准
- 医疗健康：DICOM
- 智能芯片：音视频文件、Bluetooth、WIFI
- 物联网：Zigbee，Bluetooth、WIFI、XML-SOAP、MQTT
- 金融：FIX，OpenSSL
- 视频监控：RTSP、SIP、RTP、HTTP

总结

- Fuzzing技术优势：
  （1）黑盒测试，适用范围更广泛；
  （2）动态执行，基于交互的测试，低误报率；
  （3）原理简单，便于实现；
  （4）自动化测试，无需人工参与，测试效率高。
- 专注于挖掘0-day漏洞，帮助用户做到提早预防。
- 未来发展方向：
  - 协议模型构建自动化程度的提高
  - 测试用例生成技术的研究
  - Fuzzing与白盒测试结合，实现代码定位的灰盒测试。

Geekbang 极客邦科技 InfoQ

THANKS!