OFFSEC

www.offsec.ir

نقش زبان در پیدایش و رخداد آسیب‌پذیری‌ها

امیر رسولی

amir@offsec.ir

twitter.com/Mizerium

[OFFSEC معرفی]

امیر رسولی

پایه‌گذار و مدیر تحقیقات آفسک (۱۳۹۲ تا به حال)

- امنیت فناوری‌های پیشرفته مبتنی بر وب
- امنیت زیرساخت‌های نرم‌افزاری و مهندسی معکوس کد
- امنیت سرویس‌ها و پروتکل‌های رایانه‌ای و شبکه‌ای

[هدف؛ واکاوی سوالات]

- سیستم‌ها چگونه آسیب‌پذیر می‌شوند؟
  - نقش زبان در عاملیت حملات چیست؟
- الگوپذیری این خطاها چگونه است؟ محدودیت‌های تحلیل چیست؟

[سیستم‌ها چگونه آسیب‌پذیر می‌شوند؟]

[آسیب‌پذیری سیستم‌ها؛ منوط به چند نوع]

- آسیب‌پذیری‌های ذاتی زبانی
  - آسیب‌پذیری‌های مبتنی بر ساخت پویا و نا صریحی (Implicity)
  - آسیب‌پذیری‌های مبتنی بر نقص‌های داده‌ای
  - آسیب‌پذیری‌های مبتنی بر رفتارهای تعریف‌نشده
  - نیمه‌ذاتی: آسیب‌پذیری‌های لکه‌دار (Taint-style)
- آسیب‌پذیری‌های خارجی زبانی
  - آسیب‌پذیری‌های مبتنی بر کتابخانه‌ها و الحاقات خارجی زبانی
- آسیب‌پذیری‌های نقاط تعامل زبانی

[آسیب‌پذیری‌های ذاتی زبانی]

[آسیب‌پذیری‌های مبتنی بر ساخت پویا و نا صریحی]

- ساخت پویا
  - پویایی داده‌ای
  - ساختارها و آرایه‌های پویا
  - ثابت‌های پویا
  - توابع پویا
  - کدها و فراخوانی‌های پویا

```php
<?php
    $a = "3";
    $b = 20;
    echo $a += $b; // 23
?>
```

توسعه‌ی سریع و آسان؛ اما همراه با پیچیدگی تحلیل و هزینه‌های بحرانی امنیتی

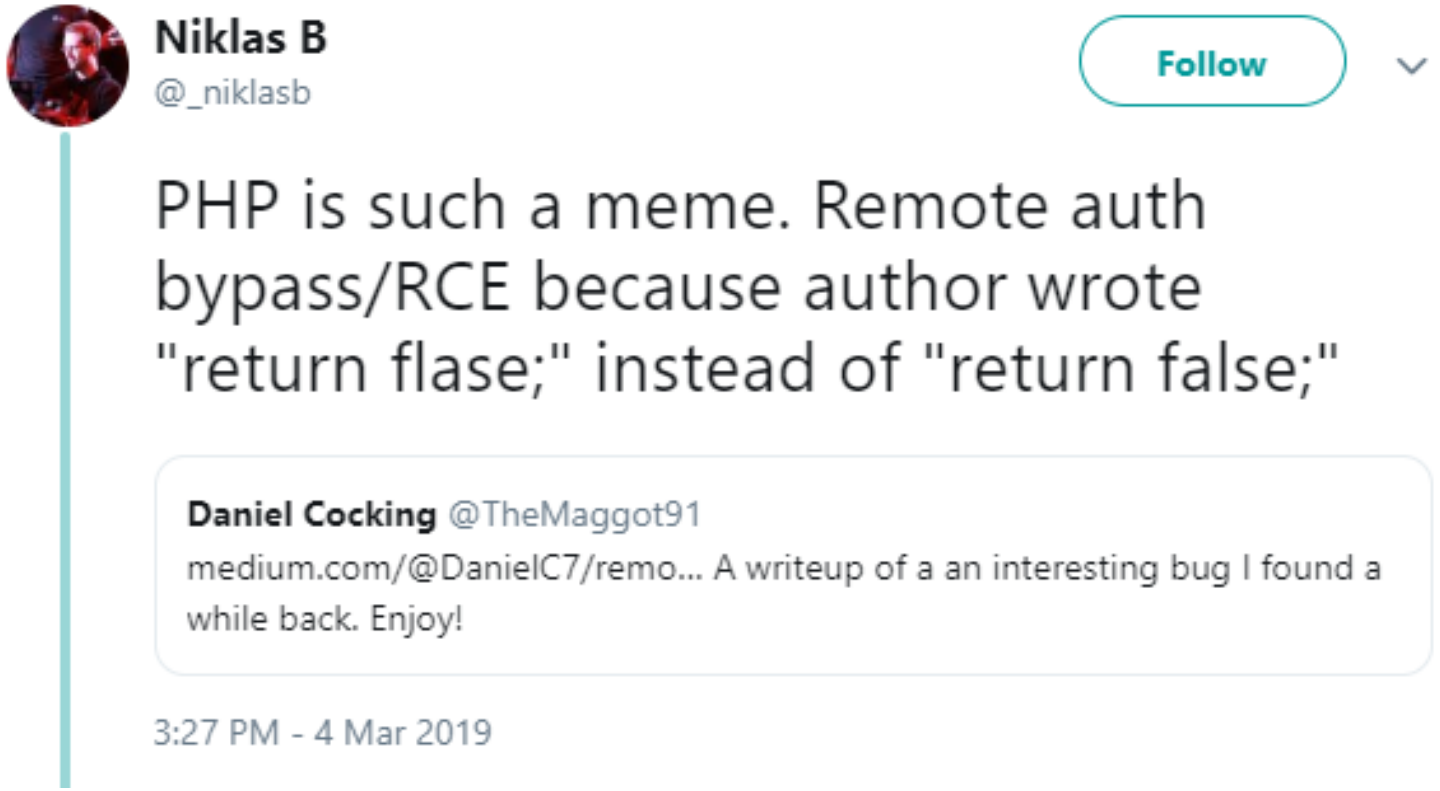

php.net
@official_php
Follow
Replying to @kadler_ibm
This is PHP, surely you didn't expect any sort of internal logical consistency.
6:04 PM - 16 Jul 2018
18 Retweets 44 Likes

[آسیب‌پذیری‌های مبتنی بر رفتارهای تعریف‌نشده]

- بیشترین اندازه آرایه‌ها در **PHP**
- متغیرهای کامپایل‌شده (**CV**) در **PHP**
- سرور **Postgres**
- و...

```php
<?php
    $a = 1;
    $c = $a + $a++;
    var_dump($c); // int(3)

    $a = 1;
    $c = $a + $a + $a++;
    var_dump($c); // int(3)
?>
```

در زبان‌هایی مانند **C** و **C++** در زمان کامپایل، این رفتارهای ناشناخته ممکن است منجر به بهینگی شود (در زبان‌های سطح بالا به ندرت اینگونه است) اما هزینه‌های امنیتی زیادی بهمراه دارد.



مثال: بروز آسیب‌پذیری دورزدن مکانیزم احراز هویت و اجرای دستور از راه دور در **PHP** براساس رفتار ناشناخته با اشتباه در یک کلمه (**flase** و **false**)

```
SELECT ((-9223372036854775808)::int8)/(-1);
```

مثال: در سرور پایگاه‌داده **Postgres** چنانچه این پرس‌وجوی **SQL** اجرا شود باعث سرریز بافر عدد صحیح علامتدار خواهد شد. بدلیل بروز رفتارهای ناشناخته در نسخه‌های ۳۲ و ۶۴ بیتی سیستم‌عامل‌های ویندوز، این حمله روی نسخه‌های ۶۴ بیتی انجام می‌شود.

[آسیب‌پذیری‌های مبتنی بر نقص‌های داده‌ای]

- ایمنی حافظه (Memory Safety)
  - مساله‌ی بازیافت حافظه

```
void vuln(char *u1) {

    /* assert(strlen(u1) < MAX); */
    char tmp[MAX];
    strcpy(tmp, u1);
    return strcmp(tmp, "foo");

}
```

Command	Description
$x \leftarrow e$	Local assignment
$x \leftarrow [e]$	Read from heap
$[e_1] \leftarrow e_2$	Heap assignment
$x \leftarrow \text{alloc}(e_{size})$	Allocation
$\text{free}(e)$	Deallocation
skip	Do nothing
if e then c_1 else c_2	Conditional
while e do c end	Loop
$c_1; c_2$	Sequencing

[آسیب‌پذیری‌های مبتنی بر نقص‌های داده‌ای]

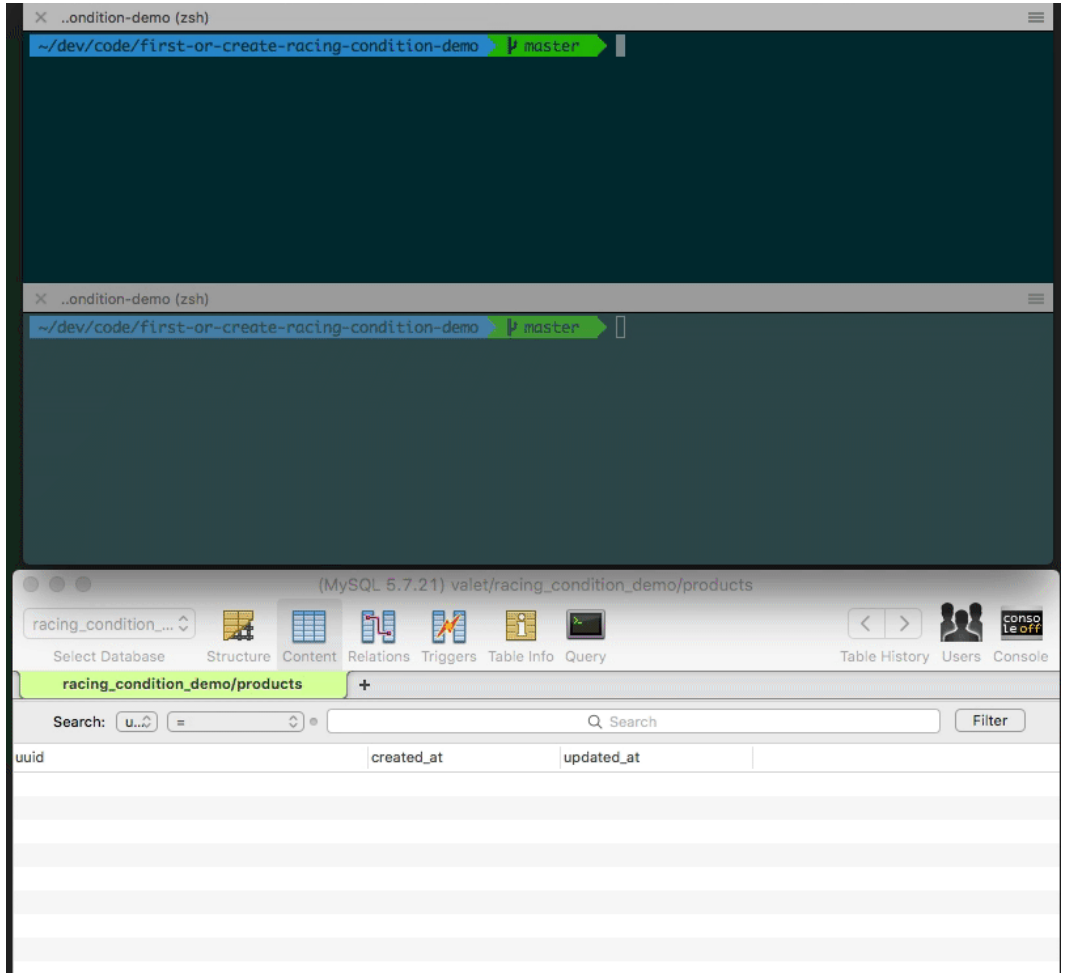
- ایمنی نوع داده‌ای (Type Safety)

```
void alloc(int num) {

        #define MAX_CHUNKS 10
        if (num > MAX_CHUNKS)
                // throw error

        void* chunks=malloc(num*sizeof(CHUNK));
        ...

};
```

نوع داده‌ای int بعنوان یکی از پرکاربردترین و معمول‌ترین انواع داده‌ای

(سرریز حافظه از نوع عدد صحیح در شرایط مقدار علامت‌دار)

[آسیب‌پذیری‌های مبتنی بر نقص‌های داده‌ای]



- مسابقه‌های داده‌ای (Data Races)

و ایمنی همزمانی

(Concurrency/Thread Safety)

مثال: چارچوب نرم‌افزاری Laravel

بعنوان یکی از فریم‌ورک‌های مطرح PHP

[آسیب‌پذیری‌های لکه‌دار]

- تزریق دستوری
- **PRNG Seed** ناقص
- پاکسازی (**Sanitization**) ناقص
- محدودیت‌های بافری
- و...

```
Stream Wrappers

SSRF: file_get_contents($_GET['url']);
RFI: include($_GET['news']);
XXE: <! ENTITY xxe SYSTEM "http://offsec.ir">
```

توسعه‌ی سریع و آسان؛ اما همراه با پیچیدگی تحلیل و هزینه‌های امنیتی

[آسیب‌پذیری‌های لکه‌دار]

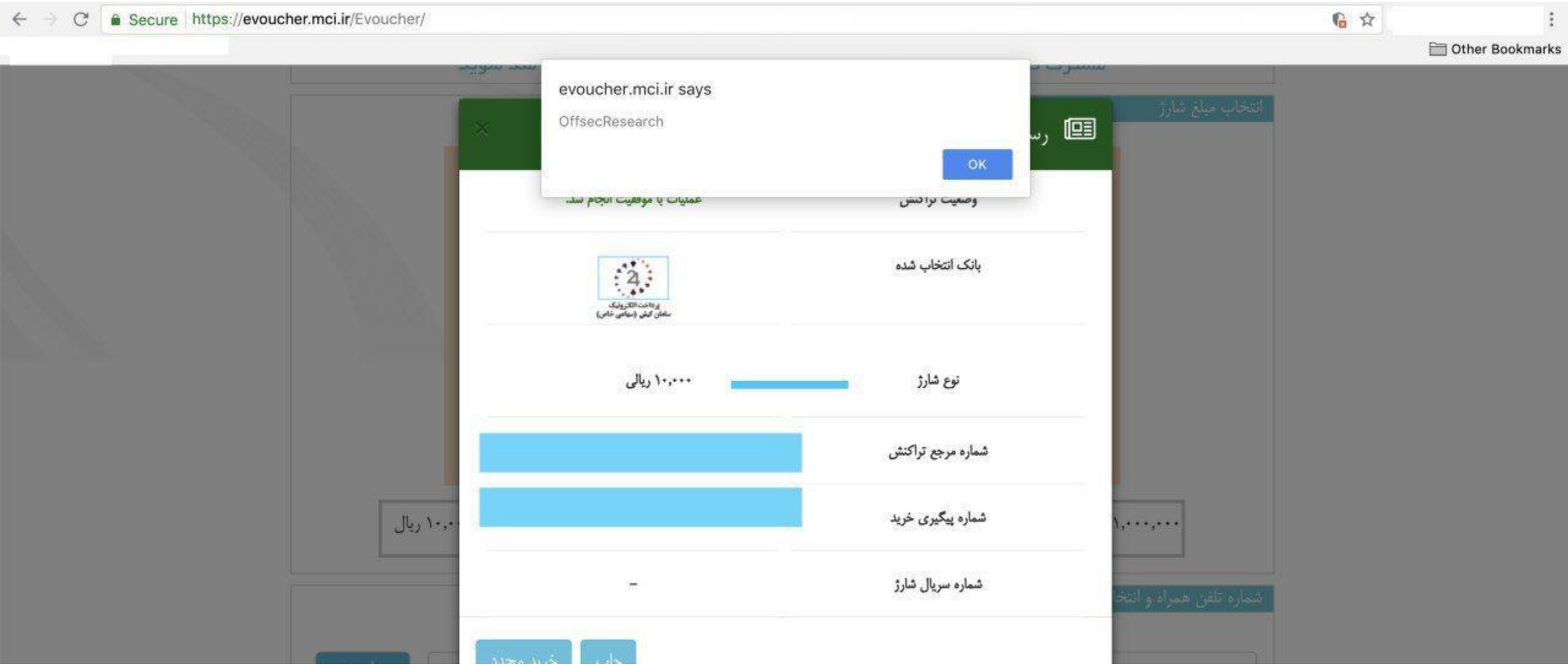
- منابع کنترل‌پذیر از سمت مهاجم
  - منابع فایلی و شبکه‌ای

(مانند توابع recv()، recvfrom() ، include() و...)

- وان (**Sink**) های حساس
  - توابع ناامن کتابخانه‌ها

(مانند strcpy(), memcpy() و system())

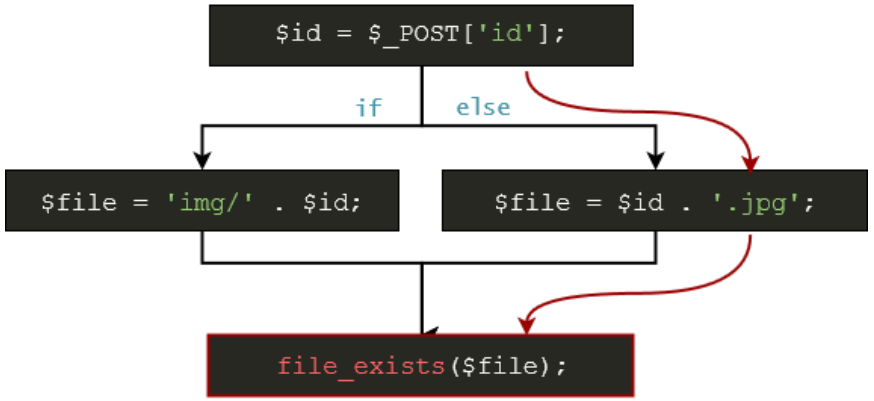
- مسیرهای داده‌ای
  - جریان اشاعه داده بین یک نقطه ورودی و یک وان حساس



مثال: نقص پاکسازی فراخوان (Callback Sanitization Flaw)

```
00000000  3C 3F 70 68 70 20 5F 5F 48 41 4C 54 5F 43 4F 4D  <?php.__HALT_COM
00000010  50 49 4C 45 52 28 29 3B 20 3F 3E 0D 0A 5F 00 00  PILER();.?>.._..
00000020  00 01 00 00 00 11 00 00 00 01 00 00 00 00 00 29  ...............)
00000030  00 00 00 4F 3A 38 3A 22 41 6E 79 43 6C 61 73 73  ...O:8:"AnyClass
00000040  22 3A 31 3A 7B 73 3A 34 3A 22 64 61 74 61 22 3B  ":1:{s:4:"data";
00000050  73 3A 34 3A 22 72 69 70 73 22 3B 7D 08 00 00 00  s:4:"rips";}....
00000060  74 65 73 74 2E 74 78 74 04 00 00 00 5D C5 6E 5B  test.txt....]┼n[
00000070  04 00 00 00 C7 A7 8B 3B B6 01 00 00 00 00 00 00  ....╟°ï;╢.......
00000080  74 65 78 74 E9 E9 6A 7A 90 17 91 F2 23 E5 FB 8D  textΘΘjzÉ.æ≥#σ√ì
00000090  DC DE 2A 60 D4 8F 7F 88 02 00 00 00 47 42 4D 42  ▄▌*`╘Å⌂ê....GBMB
```

Sam Thomas – Secarma Labs

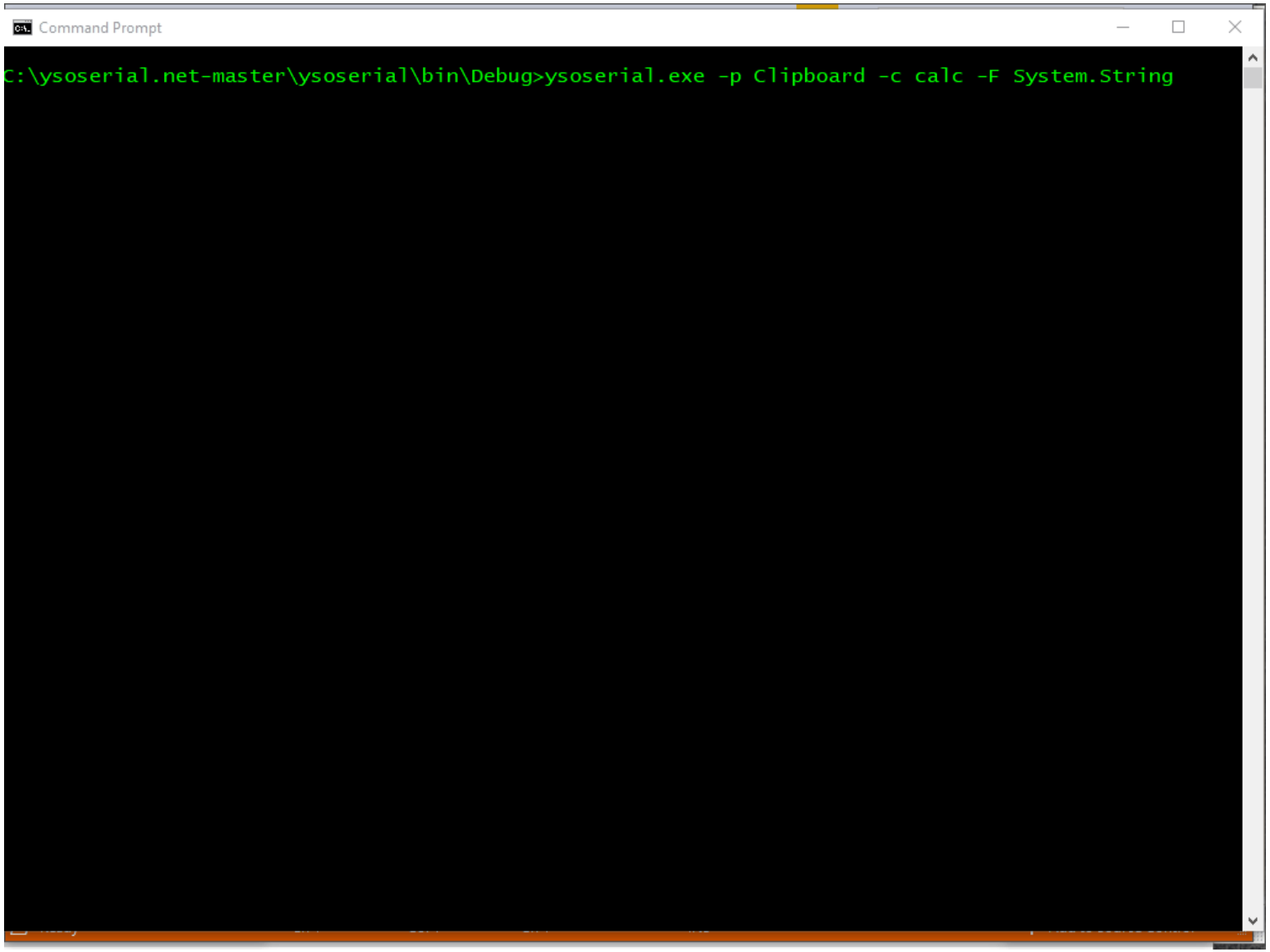


مثال: نقص برگردان سریال‌سازی اشیا در PHP (Phar Deserialization)

Sam Thomas – Secarma Labs

```
// If the file is relative, prepend upload dir.
if ( $file && 0 !== strpos( $file, '/' ) && ! preg_match( '|^.:\\\|', $file ) &&
( ( $uploads = wp_get_upload_dir() ) && false === $uploads['error'] ) )

{ $file = $uploads['basedir'] . "/$file"; }

function wp_get_attachment_thumb_file( $post_id = 0 ) {
$post_id = (int) $post_id;
if ( !$post = get_post( $post_id ) )
return false;
if ( !is_array( $imagedata = wp_get_attachment_metadata( $post->ID ) ) )
return false;
$file = get_attached_file( $post->ID );
if ( !empty($imagedata['thumb']) &&
($thumbfile = str_replace(basename($file), $imagedata['thumb'],

$file)) && file_exists($thumbfile) ) {
```

مثال: نقص سریال‌سازی اشیا در **PHP** (سامانه مدیریت محتوای Wordpress بعنوان مشهورترین **CMS** برپایه **PHP**)



مثال: نقص سریال‌سازی اشیا در **.Net**

Heartbleed

```
1  /* ssl/t1_lib.c */
2  // [...]
3  int tls1_process_heartbeat(SSL *s) {
4    unsigned char *p = &s->s3->rrec.data[0], *p1;
5    unsigned short hbtype;
6    unsigned int payload;
7    unsigned int padding = 16; /* Use minimum padding */
8    /* Read type and payload length first */
9    hbtype = *p++;
10   n2s(p, payload);
11   p1 = p;
12   // [...]
13   if (hbtype == TLS1_HB_REQUEST) {
14     unsigned char *buffer, *bp;
15     int r;
16     buffer = OPENSSL_malloc(1 + 2 + payload + padding);
17     bp = buffer;
18     /* Enter response type, length and copy payload */
19     *bp++ = TLS1_HB_RESPONSE;
20     s2n(payload, bp);
21     memcpy(bp, p1, payload);
22     bp += payload;
23     /* Random padding */
24     RAND_pseudo_bytes(bp, padding);
25     r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer,
26                          3 + payload + padding);
27     // [...]
28   }
29   // [...]
30   return 0;
31 }
```

مثال: آسیب‌پذیری بحرانی بیش‌خوانی بافر (Buffer Over-read) در OpenSSL

[آسیب‌پذیری‌های خارجی زبانی]

RARLAB WinRAR®

CVE-2018-20250

Whats New

21. Nadav Grossman from Check Point Software Technologies informed us about a security vulnerability in UNACEV2.DLL library. Aforementioned vulnerability makes possible to create files in arbitrary folders inside or outside of destination folder when unpacking ACE archives.

    WinRAR used this third party library to unpack ACE archives. UNACEV2.DLL had not been updated since 2005 and we do not have access to its source code. So we decided to drop ACE archive format support to protect security of WinRAR users.

    We are thankful to Check Point Software Technologies for reporting this issue.

مثال: آسیب‌پذیری بحرانی ساخت فایل ناخواسته در **WinRar** (در یک کتابخانه خارجی)

CVE-2019-6977

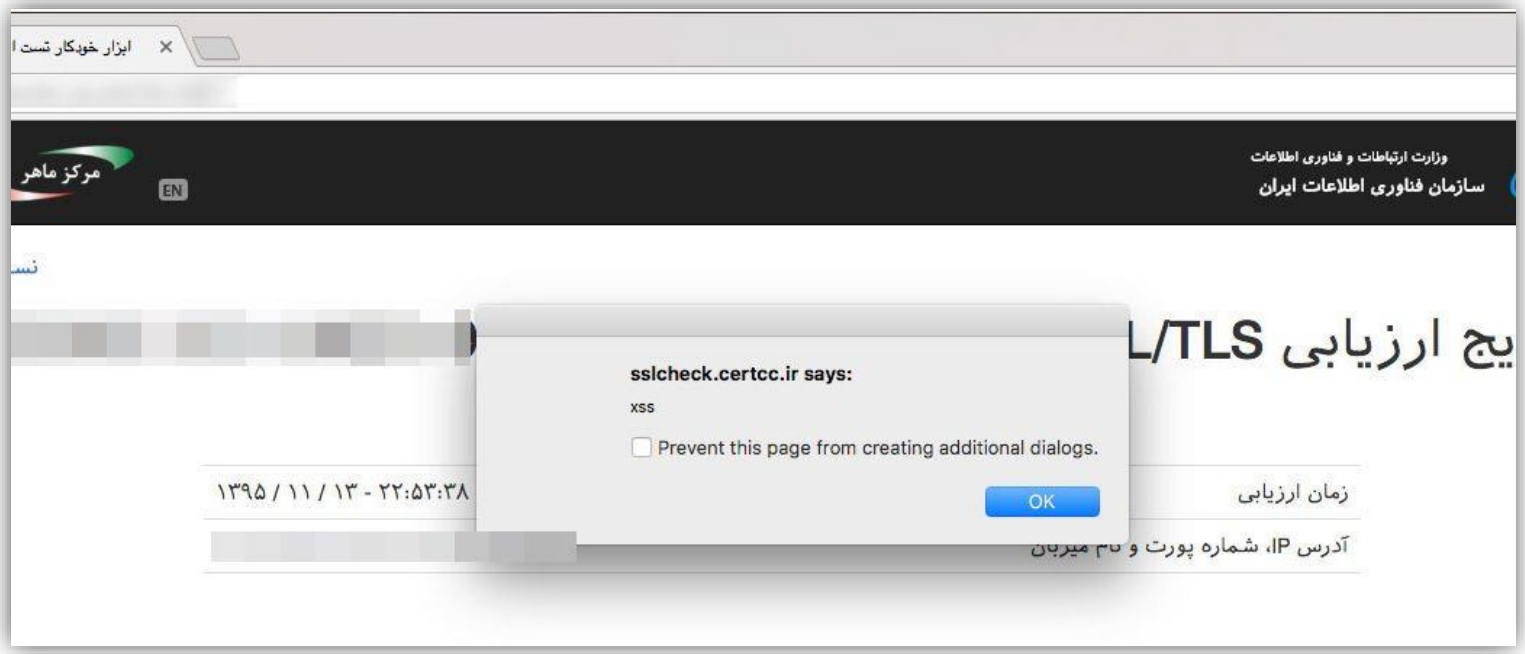
```php
function wp_crop_image( $src, $src_x, $src_y, $src_w, $src_h, $dst_w, $dst_h, $src_abs = false, $dst_file = false ) {
        $src_file = $src;
        if ( is_numeric( $src ) ) { // Handle int as attachment ID
                $src_file = get_attached_file( $src );

                if ( ! file_exists( $src_file ) ) {
                        // If the file doesn't exist, attempt a URL fopen on the src link.
                        // This can occur with certain file replication plugins.
                        $src = _load_image_to_edit_path( $src, 'full' );
                } else {
                        $src = $src_file;
                }
function get_attached_file( $attachment_id, $unfiltered = false ) {
        $file = get_post_meta( $attachment_id, '_wp_attached_file', true );
```

مثال: آسیب‌پذیری بحرانی سرریز بافر **Heap** در **PHP** (در یک کتابخانه خارجی به نام **LibGD**)

مورد استفاده در **Wordpress** جهت **Crop** کردن تصاویر (حمله بارگذاری فایل‌های مخرب)

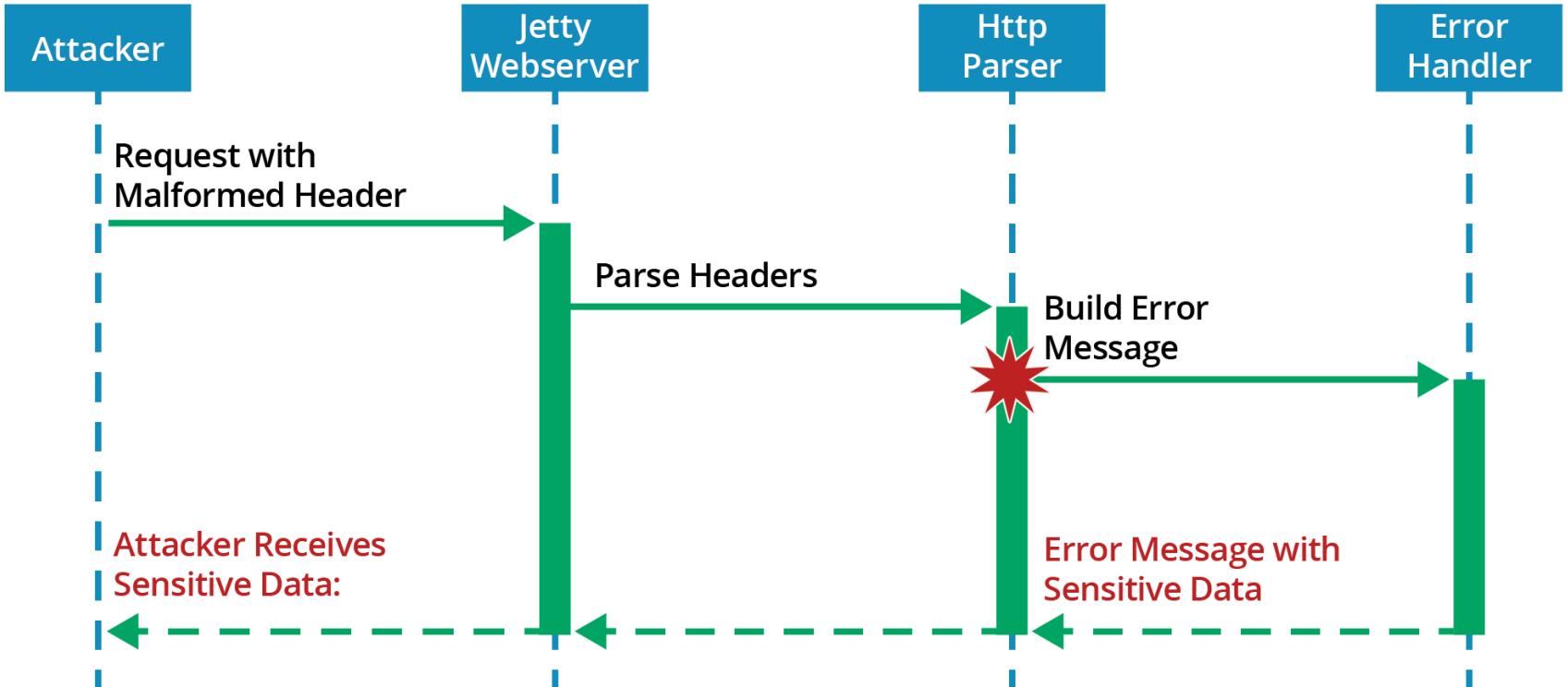
[آسیب‌پذیری‌های نقاط تعامل زبانی]

[آسیب‌پذیری‌های نقاط تعامل زبانی]



Babak Aminazad – Offsec Research

- سرویس‌های میانجی
  - یکپارچه‌سازی
  - اعمال فیلترینگ
- خدمات‌دهی به کاربران خارج از محدوده
- احراز هویت



مثال: آسیب‌پذیری بحرانی افشای اطلاعات در وب سرور برپایه جاوا (**Jetty**) در بین سرویس هندل کننده خطاها و پارسر **HTTP**

محتوای بافر بعد از اولین درخواست HTTP :

محتوای بافر بعد از درخواست دوم HTTP :

کران بالا در خواندن: آخرین ناحیه نوشته شده

مثال: آسیب‌پذیری بحرانی افشای اطلاعات در وب سرور برپایه جاوا (Jetty) در بین سرویس هندل کننده خطاها و پارسرHTTP (شمای شبیه سازی شده طراحی حمله Out of Bounds)

phpWhois implements multiple generic parsers for WHOIS data in `whois.parser.php`. The parser implemented in function `generic_parser_b` is vulnerable to injection of PHP code.

The function `generic_parser_b` builds a PHP statement from WHOIS data values by concatenating strings without proper sanitization. It then passes the statement to the `eval` function:

```
function generic_parser_b($rawdata, $items = array(), $dateformat = 'mdy', $hasreg = true, $scanall = false) {
[...]
    foreach ($rawdata as $val) {
        if (trim($val) != '') {
            if (($val[0] == '%' || $val[0] == '#') && $disok) {
                $r['disclaimer'][] = trim(substr($val, 1));
                $disok = true;
                continue;
            }
            $disok = false;
            reset($items);
            foreach ($items as $match => $field) {
                $pos = strpos($val, $match);
                if ($pos !== false) {
                    if ($field != '') {
                        $var = '$r' . getvarname($field);
                        $itm = trim(substr($val, $pos + strlen($match)));
                        if ($itm != '')
                            eval($var . '="' . str_replace('"', '\"', $itm) . '";');
                    }
```

مثال: آسیب‌پذیری بحرانی تزریق کد در یک طراحی سرویس خارجی دریافت کننده

(و پارسر) اطلاعات دامنه

[الگوپذیری آسیب‌پذیری‌ها و محدودیت‌های تحلیل]

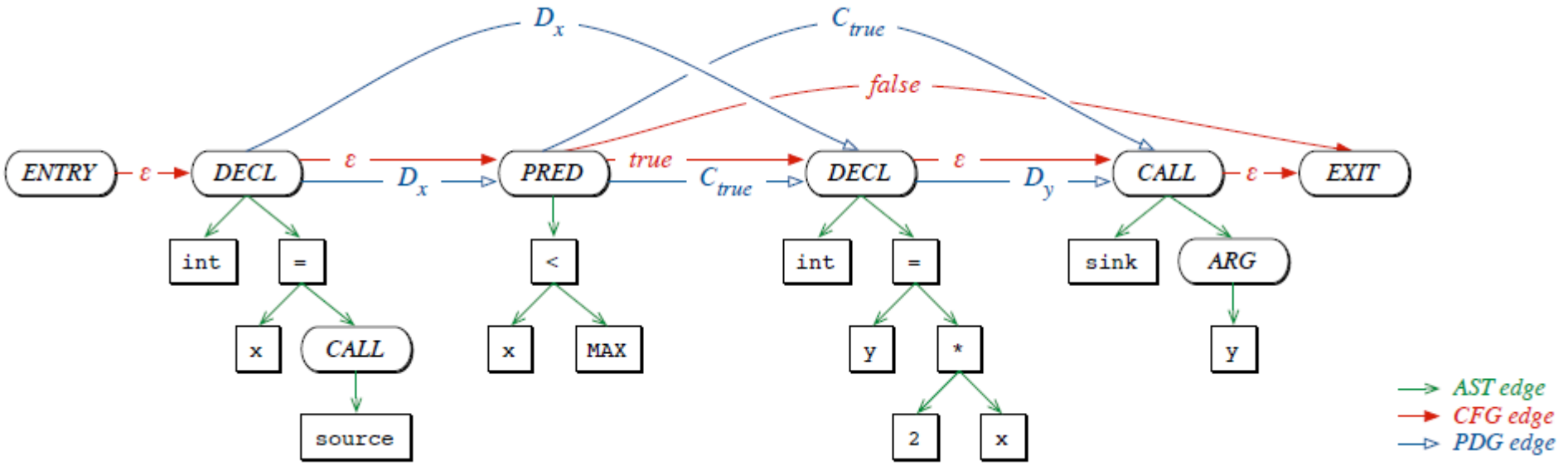
[الگوپذیری آسیب‌پذیری‌ها]

- مدلی که
  - بتواند آسیب‌پذیری‌ها را با الگوی ثابت آنان بشناسد
  - بعنوان یک عنصر یادگیرنده برای الگوریتم یادگیری ماشین بکار گرفته شود
    - سرعت و دقت تحلیل را بهمراه بیاورد
  - کمترین میزان اشتباهات (**False Positives**) و سربار را داشته باشد
  - بیشترین میزان دقت را داشته باشد و از پیچیدگی‌های تحلیل بکاهد

[الگوپذیری آسیب‌پذیری‌ها]

- مدل‌های تحلیل ایستا
  - درخت نحو انتزاعی (**Abstract Syntax Tree**)
  - گراف جریان کنترلی (**Control Flow Graph**)
  - گراف وابستگی برنامه (**Program Dependence Graph**)

Towards Pattern-Based Vulnerability Discovery in Binary Code – F. Yamaguchi



گراف خاصیت کد (**Code Property Graph**) دکتر یاماگوچی، این امکان را در اختیار تحلیلگر می‌گذارد که با ادغام و روی هم قرار دادن بازنمایانه‌ی گراف‌های برشمرده‌شده، بتواند آسیب‌پذیری‌های سطح باینری را (بغیر از Race Condition ها و خطاهای طراحی) باتوجه به الگوی آنان در مدل‌های AST، CFG و PDG کشف کند. ابزار Joern نمونه متن‌باز پیاده‌سازی‌شده این مدل در C/C++ است.

[الگوناپذیری آسیب‌پذیری‌ها]

- مسابقه‌های داده‌ای (وابسته به امنیت همزمانی سیستم‌عامل‌ها و اجزای بیرونی)
- آسیب‌پذیری‌های مبتنی بر اجزای داده‌ای خارجی (**Out of Bounds**)
- آسیب‌پذیری‌های تولید شده‌ی پویا و یا مشمول مبهم‌سازی (**Obfuscation**)
- اثرهای متقابل اجزای سیستم بر نقص‌ها و آسیب‌پذیری‌های طراحی
- خطاهایی که در زمان اجرا بر روند معمول برنامه حادث می‌شوند و آسیب‌پذیری دیگری بوجود خواهند آورد

[محدودیت‌های تحلیل]

- محدودیت‌های مدل تحلیل ایستا
  - لزوم در اختیار داشتن منابع کد
  - الگوناپذیری برخی آسیب‌پذیری‌ها

- محدودیت‌های مدل تحلیل پویا
  - عدم شناسایی و محدودیت در اشراف بسیاری از آسیب‌پذیری‌ها بدلیل عدم درک سیستم و چگونگی کارکرد آن