浏览器1 day攻防与检测

LCatro

唯品会安全应急响应中心

目录

CONTENTS

01 常见的攻击方式

点击添加你的副标题

常见的攻击方式

- 钓鱼攻击 (针对邮件与特定用户的攻击)
- 网页挂马 (挂黑页与水坑攻击)
- XSS (不需要渗透到主机修改代码)
- Wifi 劫持 (请移步到KFC 连接Wifi)

02 常见的漏洞与漏洞原理

浏览器常见漏洞

- 远程代码执行
- UXSS

浏览器远程代码执行

- 二进制(浏览器内核与JavaScript引擎)
- 浏览器插件(二进制漏洞与接口设计问题)
- Android 敏感接口泄漏(跨域调用特权域API)

浏览器远程代码执行

- 二进制(浏览器内核与JavaScript引擎)
- 浏览器插件(二进制漏洞与接口设计问题)
- Android 敏感接口泄漏(跨域调用特权域API)

浏览器远程代码执行漏洞之Android敏感接口泄漏

- WebView 与addJavascriptInterface()

  addJavascriptInterface() 添加Java 代码到WebView 作为JavaScript 函数和类调用

Link : http://www.loner.fm/bugs/bug_detail.php?wybug_id=wooyun-2016-0187345

- intent 和伪协议

  利用intent 和伪协议触发第三方APP 和应用程序启动

Link : http://www.loner.fm/bugs/bug_detail.php?wybug_id=wooyun-2016-0194107

一个简单的崩溃演示

- IE 11 远程拒绝服务0 day

```
var crash = document.creat
crash
```

浏览器UXSS

- UXSS 与XSS 的区别

  XSS :WEB 后台过滤不严格,导致用户的输入可以被浏览器执行;

  UXSS :浏览器内部逻辑问题导致脚本跨域执行

- UXSS 的原理

  HTML 元素在浏览器中的实现存在域限制,在触发漏洞的情况下,HTML 元素中的域可以被任意修改,达到跨域脚本执行

03 利用方式

浏览器远程代码执行利用

- 漏洞缓解利用:堆风水,ASLR,DEP,CFG
- ROP
- ShellCode

浏览器UXSS利用

- document.cookie + XMLHttpRequest 任意域Cookie 获取

  在不依赖XSS 的情况下获取用户的访问权限,配合钓鱼,水坑攻击和第三方广告插件

04 漏洞检测

浏览器漏洞自动化检测的意义

- 在黑产的眼里

  钓鱼,挂马,广告流量背后的利益链

- 在厂商的眼里

  产品的安全性与用户口碑

浏览器漏洞自动化检测

- UXSS 的检测方式

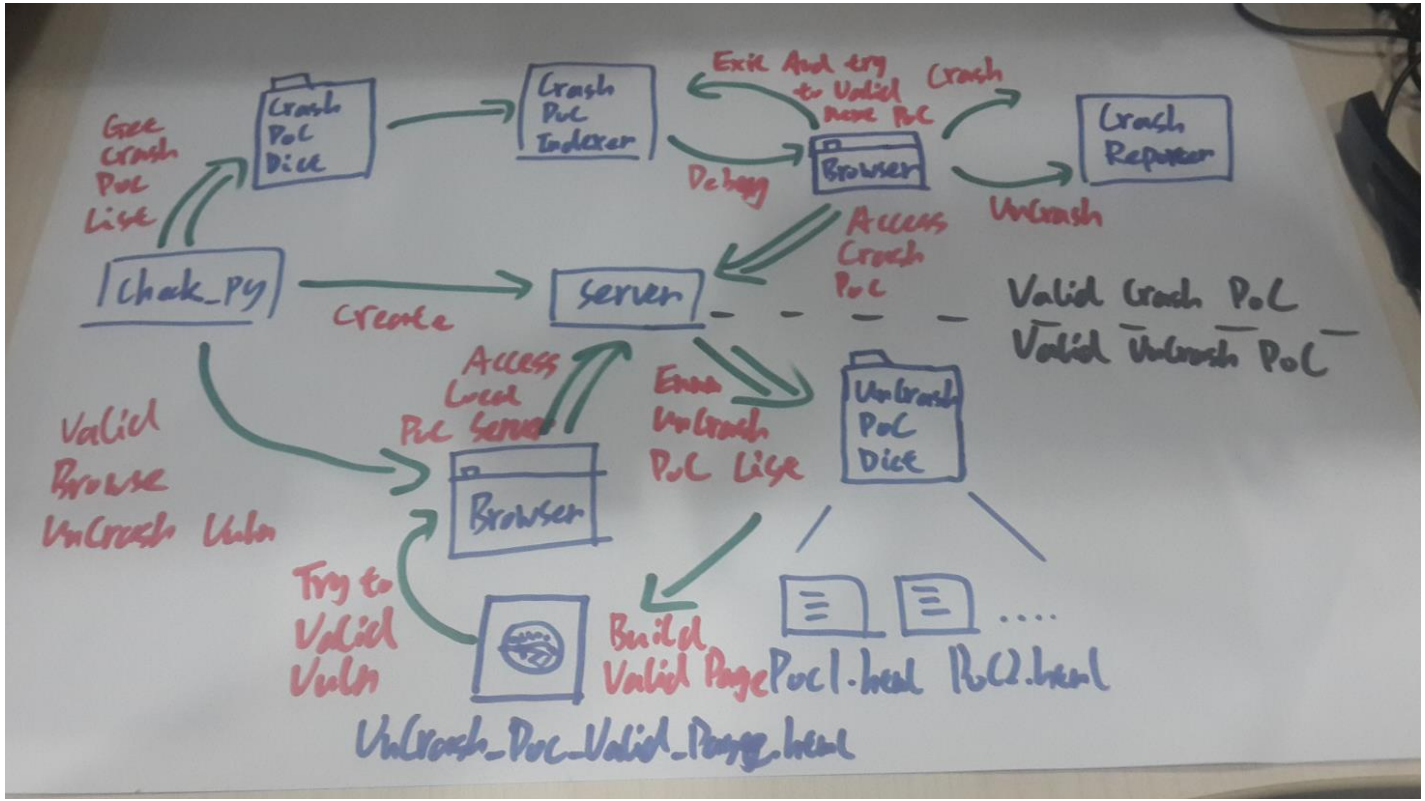
  <iframe> 与postMessage()

- 二进制漏洞的检测方式

  对象泄漏与崩溃检测

Link : https://github.com/lcatro/browser_vuln_check

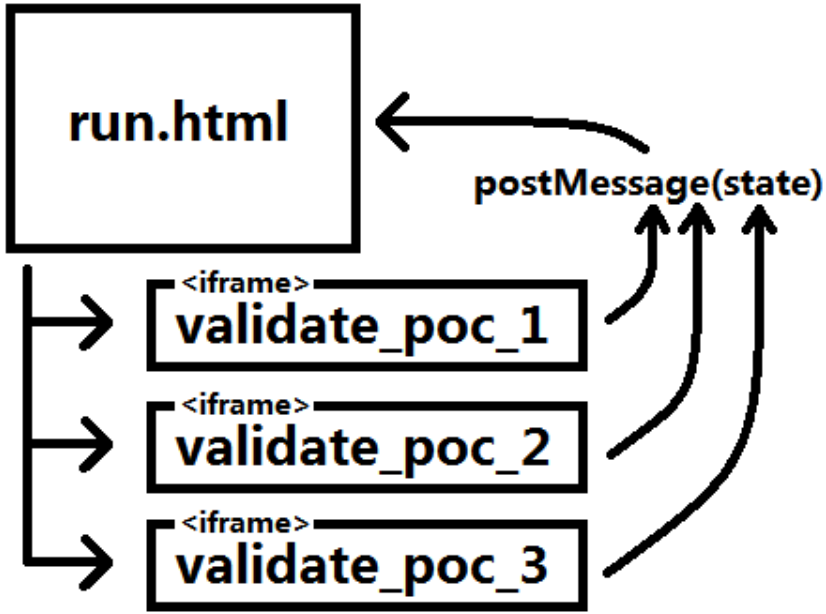
浏览器漏洞自动化检测架构



浏览器漏洞自动化检测之非崩溃漏洞检测原理

- <iframe> 的作用在于提供一个干净,隔离的新环境,防止运行多个PoC 的结果相互影响

- postMessage() 返回测试结果



浏览器漏洞自动化检测之非崩溃漏洞检测原理

- 在run.html body 标签中引入UXSS PoC

```
<body>
    <div id="output_state_window"></div>
```

```
uncrash_poc_list=list_dir_file(get_current_path()+'uncrash_poc')  ##  Read All UnCrash PoC File
uncrash_poc_enter_point_list=[]

for uncrash_poc_index in uncrash_poc_list :
    if (uncrash_poc_index[1]=='index.html' or
        0==uncrash_poc_index[1].find('CVE') or
        0==uncrash_poc_index[1].find('POC')) :  ##  Filter Main EnterPoint UnCrash CVE PoC File
        uncrash_poc_enter_point_list.append(get_relative_path(False,uncrash_poc_index[0]))

insert_to_temple_framework_code=''

for uncrash_poc_enter_point_index in uncrash_poc_enter_point_list :  ##  Build PoC Call <iframe> Code
    insert_to_temple_framework_code+='<iframe src="uncrash_poc?poc_name='+uncrash_poc_enter_point_index+'" style="visib
```

浏览器漏洞自动化检测之非崩溃漏洞检测原理

- run.html 接收从iframe 中检测完成的结果

```
        window.onmessage=function(message) {
            var result=message.data;

            report_check_state(JSON.stringify(message.data));
            print_in_window(result.vuln_name+' check_state='+result.vuln_valid_state);
            console.log(result.vuln_name+' check_state='+result.vuln_valid_state);

            poc_valid_index++;
//            console.log(poc_valid_index);

            if (poc_count==poc_valid_index)
                exit_uncrash_poc_scan();
        };
```

浏览器漏洞自动化检测之非崩溃漏洞检测PoC

- PoC BadKernel

```
VULN_NAME='bad_kernel';
VULN_VERSION=['Wechat 6.3.0-Wechar 6.3.12'];

function post_result(check_state) {
    var parentwin = window.parent;
    var post_result_json={};
    post_result_json.vuln_name=VULN_NAME;
    post_result_json.vuln_version=VULN_VERSION;
    post_result_json.vuln_valid_state=check_state;

    parentwin.postMessage(post_result_json,'*');
}

function check_vuln() {
    var kMessages;
    Object.prototype.__defineGetter__("observe_accept_invalid",function(){
        log("called");
        kMessages=this});
    try{Object.observe({},function(){},1)}catch(e){}
    delete Object.prototype["observe_accept_invalid"];

    if (undefined!=kMessages)
        post_result(true);
    else
        post_result(false);
}

check_vuln();
```

浏览器漏洞自动化检测之非崩溃漏洞检测PoC

- 信息泄漏

  检测对象中泄漏的信息

PoC

```
VULN_NAME='CVE-2016-1677';
VULN_VERSION=['Chrome 50-51','Wechat 6.3'];

function post_result(check_state) {
    var parentwin = window.parent;
    var post_result_json={};
    post_result_json.vuln_name=VULN_NAME;
    post_result_json.vuln_version=VULN_VERSION;
    post_result_json.vuln_valid_state=check_state;

    parentwin.postMessage(post_result_json,'*');
}


function check_vuln() {
    var num = new Number(10);
    Array.prototype.__defineGetter__(0,function(){
        return num;
    })
    Array.prototype.__defineSetter__(0,function(value){
    })

    var str=decodeURI("%E7%9A%84");

    if (str.charCodeAt(0).toString(16)!='7684')
        post_result(true);
    else
        post_result(false);
}
```

浏览器漏洞自动化检测之崩溃漏洞检测PoC

- 崩溃检测

PyDbg

崩溃点分析

```
def access_exception_debug_event_handle(self) :
    global current_cve_name

    current_pid=self.dbg.dwProcessId
    current_address=self.exception_address

    print current_cve_name
    print 'WARNING! PID:'+str(current_pid)+' Making a Access Exception (0xC0000005) '
    print 'Detail Report :\r\n'
    print '  Exception Address:'+str(current_address)
    print '  EAX:'+str(hex(self.get_register('EAX')))[:-1],'EBX:'+str(hex(self.get_register('EBX')))[:-1],'ECX:
    print ''
```

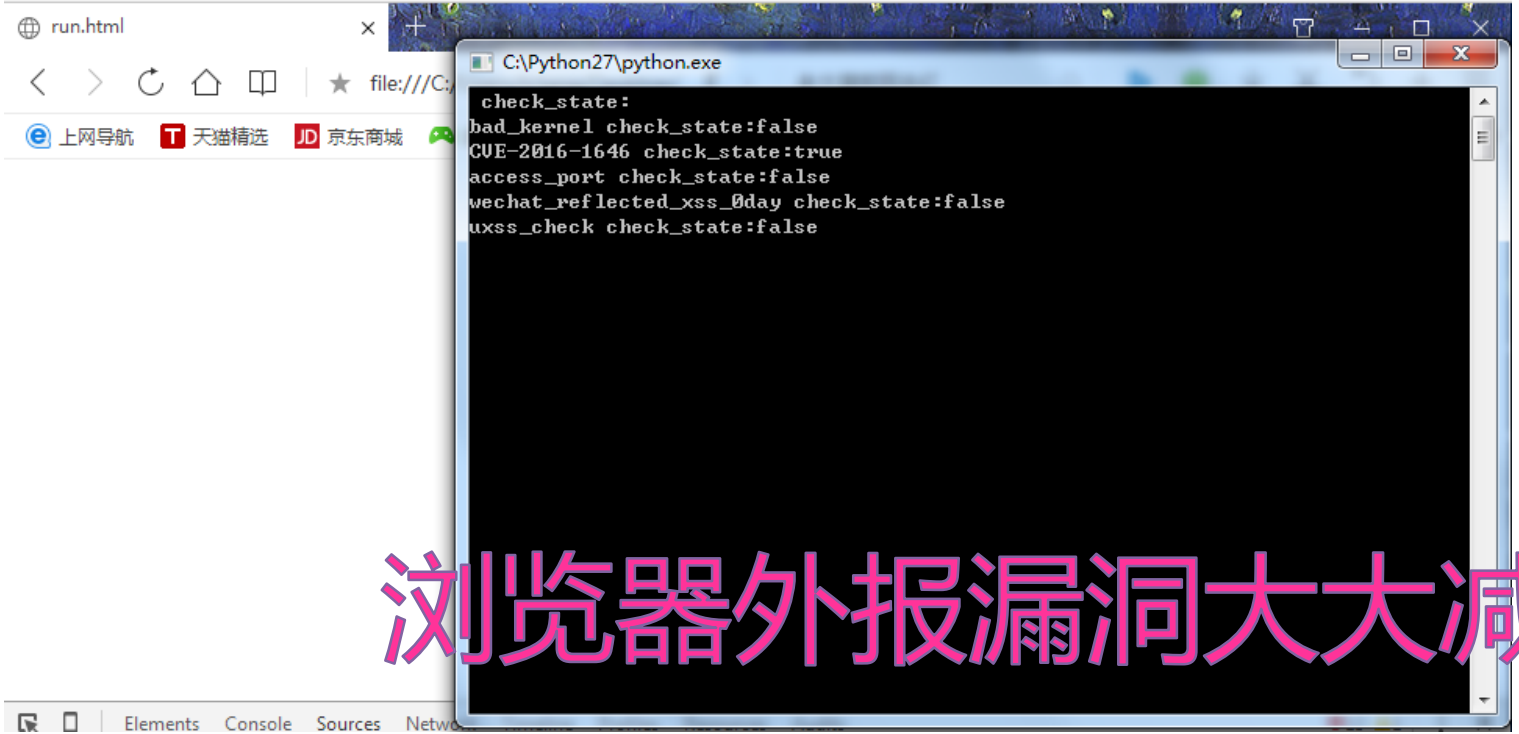
浏览器漏洞自动化检测之二进制漏洞检测

- 不容易检测的崩溃

Windows : Gflags 堆调试工具

Linux : AddressSanitizer 与valgrind 内存检测工具

浏览器漏洞自动化检测效果



Q&A

感谢聆听

—

THANKS!

SRC
唯品会安全应急响应中心