

2023-2024学年秋季学期

Web安全技术
Web Security

授课团队：刘奇旭、刘潮歌

学生助教：曹婉莹、孙承一

Web安全技术

Web Security

2.3 ClickJacking

刘奇旭

liuqixu@iie.ac.cn

中科院信工所 第六研究室



中国科学院大学
University of Chinese Academy of Sciences

一章一问

□ Clickjacking的表现形式及防御思路

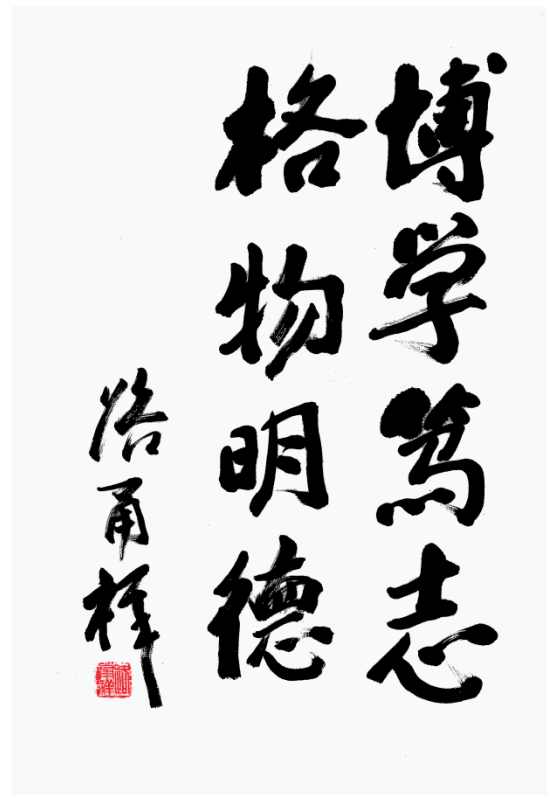


本章大纲

□ 概述

□ 攻击类型

□ 防御方法



CLICKJACKING

- ❑ Clickjacking, also known as a "UI redress attack", is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.
- ❑ Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.



A04:2021 – 不安全设计

可对照 CWEs 数量	最大发生率	平均发生率	最大覆盖范围	平均覆盖范围	平均加权漏洞	平均加权影响	出现次数	所有相关 CVEs 数量
40	24.19%	3.00%	77.25%	42.51%	6.46	6.78	262,407	2,691

CWE-256 Unprotected Storage of Credentials

CWE-257 Storing Passwords in a Recoverable Format

CWE-266 Incorrect Privilege Assignment

CWE-269 Improper Privilege Management

CWE-280 Improper Handling of Insufficient Permissions or Privileges

CWE-311 Missing Encryption of Sensitive Data

CWE-312 Cleartext Storage of Sensitive Information

CWE-313 Cleartext Storage in a File or on Disk

CWE-316 Cleartext Storage of Sensitive Information in Memory

CWE-419 Unprotected Primary Channel

CWE-430 Deployment of Wrong Handler

CWE-434 Unrestricted Upload of File with Dangerous Type

CWE-444 Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling')

举例





A04 Insecure Design

OWASP Top 10:2021

[Home](#)

[Notice](#)

[Introduction](#)

[How to use the OWASP Top 10 as a standard](#)

[How to start an AppSec program with the OWASP Top 10](#)

[About OWASP](#)

Top 10:2021 List

[A01 Broken Access Control](#)

[A02 Cryptographic Failures](#)

[A03 Injection](#)

[A04 Insecure Design](#)

[A05 Security Misconfiguration](#)

[A06 Vulnerable and Outdated Components](#)

[A07 Identification and Authentication Failures](#)

[A08 Software and Data Integrity Failures](#)

[A09 Security Logging and Monitoring Failures](#)

[A10 Server Side Request Forgery \(SSRF\)](#)

[Next Steps](#)

[CWE-579 J2EE Bad Practices: Non-serializable Object Stored in Session](#)

[CWE-598 Use of GET Request Method With Sensitive Query Strings](#)

[CWE-602 Client-Side Enforcement of Server-Side Security](#)

[CWE-642 External Control of Critical State Data](#)

[CWE-646 Reliance on File Name or Extension of Externally-Supplied File](#)

[CWE-650 Trusting HTTP Permission Methods on the Server Side](#)

[CWE-653 Insufficient Compartmentalization](#)

[CWE-656 Reliance on Security Through Obscurity](#)

[CWE-657 Violation of Secure Design Principles](#)

[CWE-799 Improper Control of Interaction Frequency](#)

[CWE-807 Reliance on Untrusted Inputs in a Security Decision](#)

[CWE-840 Business Logic Errors](#)

[CWE-841 Improper Enforcement of Behavioral Workflow](#)

[CWE-927 Use of Implicit Intent for Sensitive Communication](#)

[CWE-1021 Improper Restriction of Rendered UI Layers or Frames](#)

[CWE-1173 Improper Use of Validation Framework](#)



CWE-1021

← → ↺ cwe.mitre.org/data/definitions/1021.html



Home > CWE List > CWE- Individual Dictionary Definition (4.8)

Home | About | CWE List | Scoring | Mapping Guidance | Community | News | Search

CWE-1021: Improper Restriction of Rendered UI Layers or Frames

Weakness ID: 1021

Abstraction: Base

Structure: Simple

Presentation Filter:

Description

The web application does not restrict or incorrectly restricts frame objects or UI layers that belong to another application or domain, which can lead to user confusion about which interface the

Extended Description

A web application is expected to place restrictions on whether it is allowed to be rendered within frames, iframes, objects, embed or applet elements. Without the restrictions, users can be tricked into performing actions on the application when they were not intending to.

Alternate Terms

Clickjacking

UI Redress Attack

Tapjacking:

"Tapjacking" is similar to clickjacking, except it is used for mobile applications in which the user "taps" the application instead of performing a mouse click.

Relationships

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name
ChildOf	✓	451	User Interface (UI) Misrepresentation of Critical Information
ChildOf	✓	441	Unintended Proxy or Intermediary ('Confused Deputy')

Relevant to the view "Software Development" (CWE-699)

Nature	Type	ID	Name
MemberOf	✗	355	User Interface Security Issues

Relevant to the view "Weaknesses for Simplified Mapping of Published Vulnerabilities" (CWE-1003)

Modes Of Introduction

Phase	Note
Implementation	

Applicable Platforms

Technologies

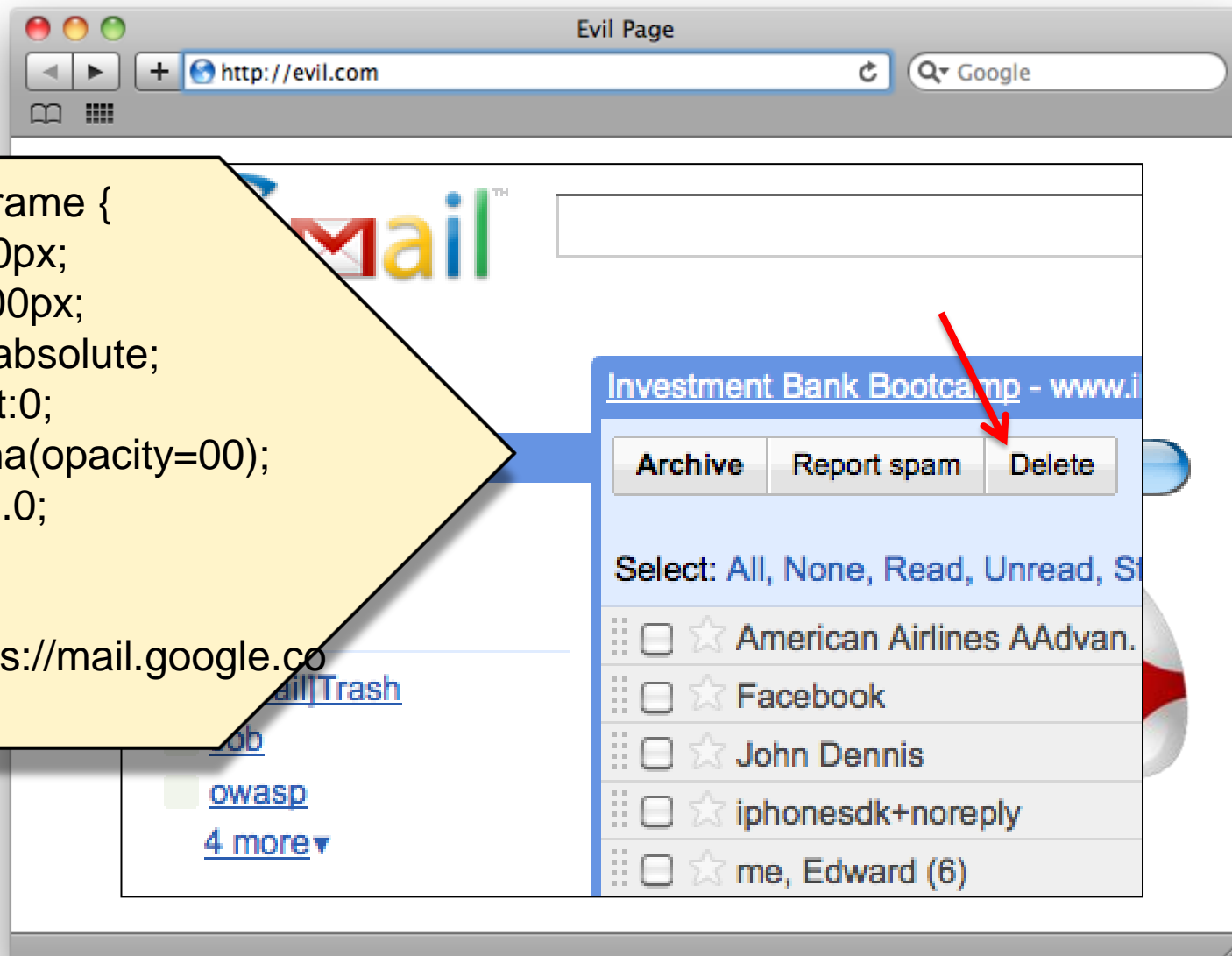
Class: Web Based (Undetermined Prevalence)



CLICKJACKING



CLICKJACKING



CLICKJACKING



CLICKJACKING

- ❑ ClickJacking称为**点击劫持攻击**，又称为**UI-覆盖攻击**。2008年由互联网安全专家罗伯特·汉森和耶利米·格劳斯曼首次提出。
- ❑ 点击劫持从根本上来说是对人类感知的攻击。在用户不知情的情况下，利用与用户产生的交互界面，诱使用户触发一些动作，达到攻击者想要实现的其它目的。



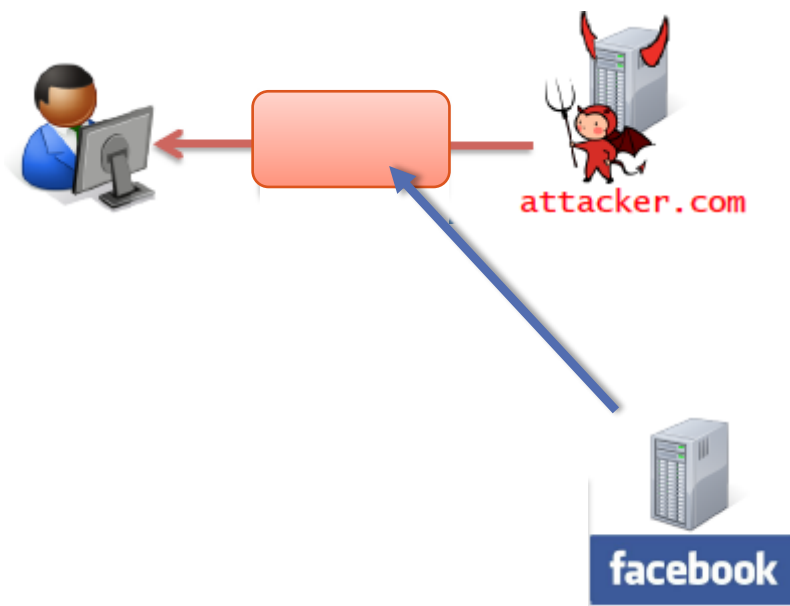
CLICKJACKING危害

- ❑ ClickJacking是一种恶意攻击技术，用于跟踪网络用户，获取其私密信息，对IE、Safari、Firefox、Opera、Adobe Flash等平台构成威胁。
- ❑ Iframe一个银行转账页面，可窃取银行账户口令，造成财产损失。
- ❑ 可能莫名其妙发出了不该发的消息，粉了不认识的人，点了一个带有黑产性质的广告，恶意刷流量。



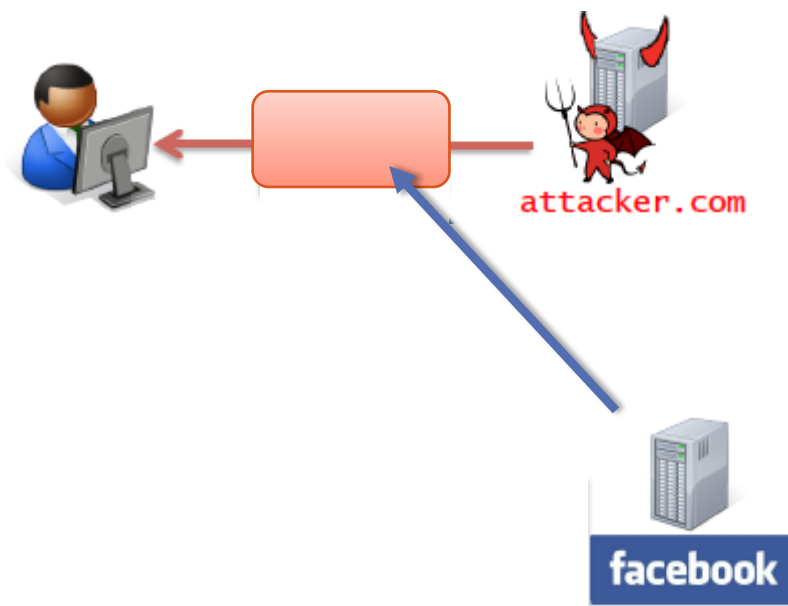
攻击原理解析

- ClickJacking是一种视觉欺骗手段，利用HTML中<iframe>标签等手段嵌套一个透明不可见的页面，让用户在不知情的情况下，点击攻击者想要欺骗用户点击的另一个置于原网页上面的透明页面。



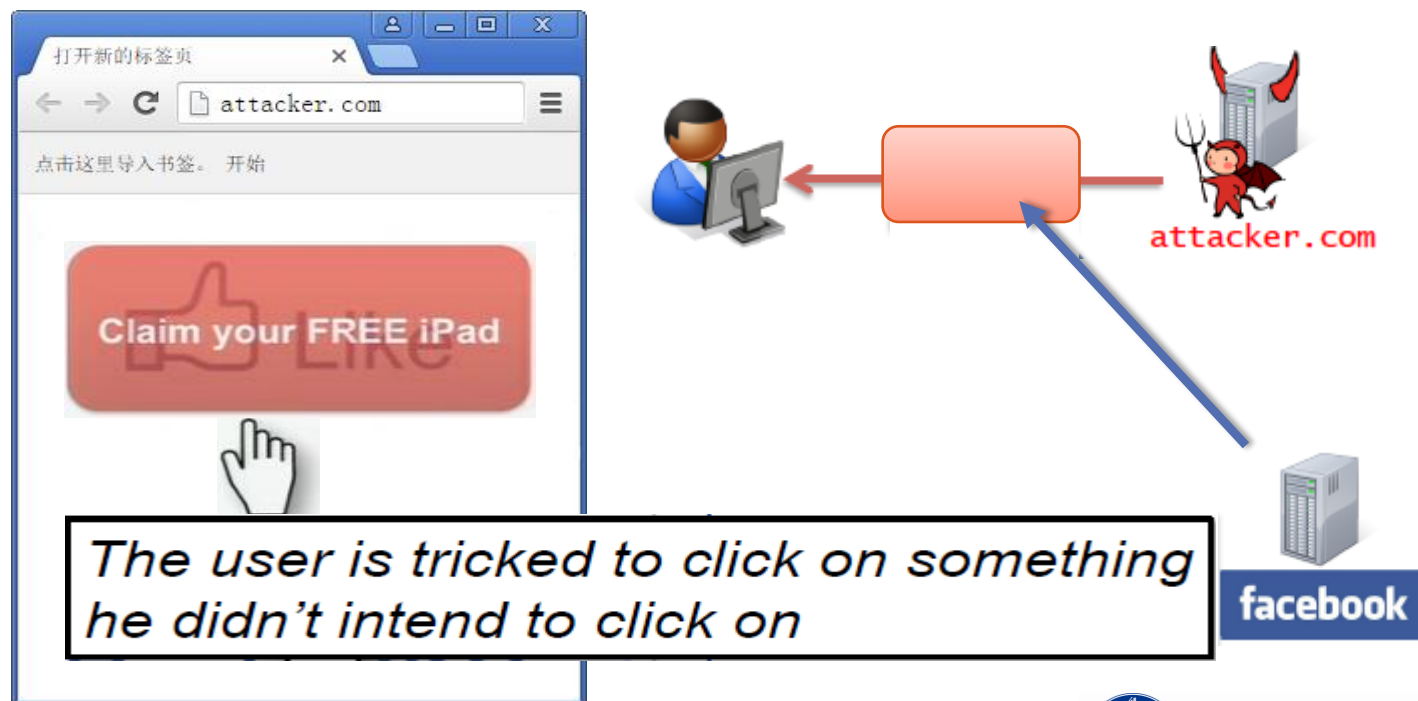
攻击原理解析

- ClickJacking是一种视觉欺骗手段，利用HTML中<iframe>标签等手段嵌套一个透明不可见的页面，让用户在不知情的情况下，点击攻击者想要欺骗用户点击的另一个置于原网页上面的透明页面。



攻击原理解析

- ClickJacking是一种视觉欺骗手段，利用HTML中<iframe>标签等手段嵌套一个透明不可见的页面，让用户在不知情的情况下，点击攻击者想要欺骗用户点击的另一个置于原网页上面的透明页面。



CLICKJACKING概述

- ClickJacking利用视觉欺骗手段，诱导用户点击被攻击者嵌入的恶意页面。
- 用户在浏览web页面时中了ClickJacking后，可能带来财产损失，账号窃取，隐私泄露，恶意吸费等。
- 实施ClickJacking，主要利用浏览器或HTML的一些特性，如：
iframe**标签透明**，image**图片浮动**等，将恶意页面覆盖原来的页面，对用户发起攻击。

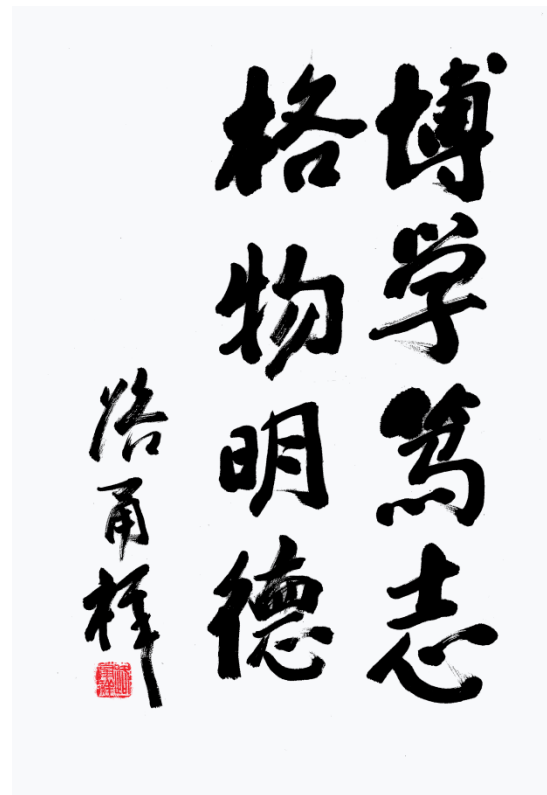


本章大纲

□ 概述

□ 攻击类型

□ 防御方法



图片覆盖劫持

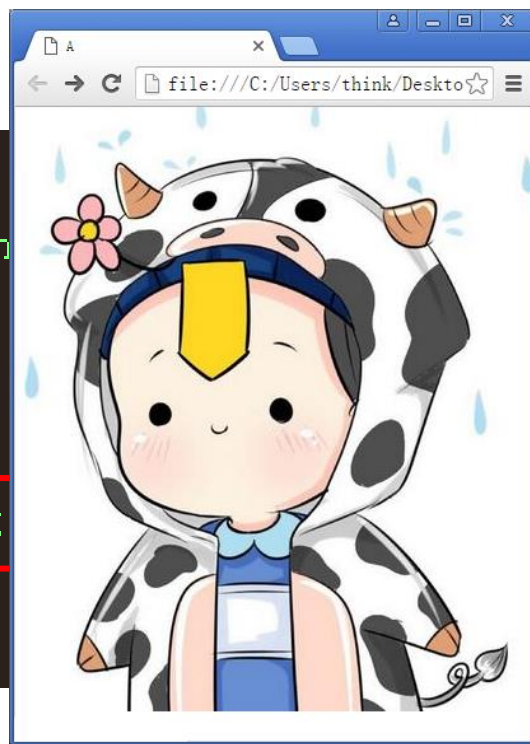
- ❑ 图片覆盖攻击（Cross Site Image Overlaying），攻击者使用一张或多张图片，利用图片的style或者能够控制的CSS，将图片覆盖在网页上，形成点击劫持。
- ❑ 当然图片本身所带的信息可能就带有欺骗的含义，这样不需要用户点击，也能达到欺骗的目的。
- ❑ 这种攻击很容易出现在网站本身的页面。



图片覆盖劫持

- 示例：在可以输入HTML内容的地方加上一张图片，该图片覆盖在指定的位置。

```
<html>
<head>
<meta http-equiv="content-type" content="text/html">
<title>A</title>
</head>
<body>
  <a href="B.html">
    This is B.
    
</body>
</html>
```

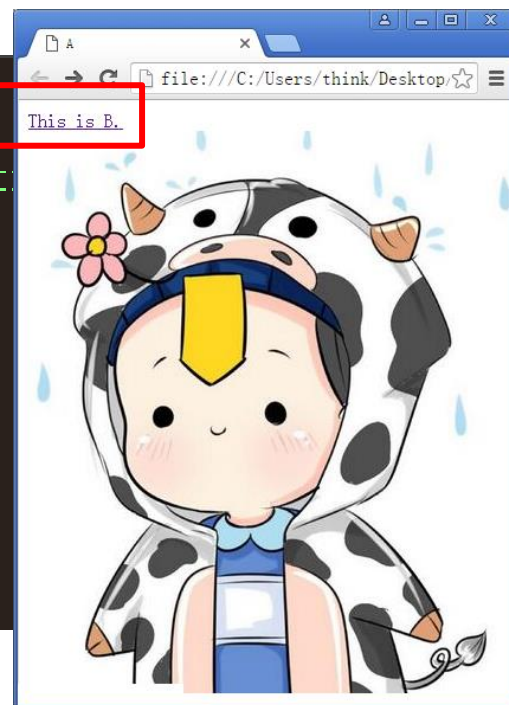


图片覆盖劫持防御

- 在防御图片覆盖攻击时，需要检查用户提交的HTML代码中，img标签的style属性是否可能导致浮动。

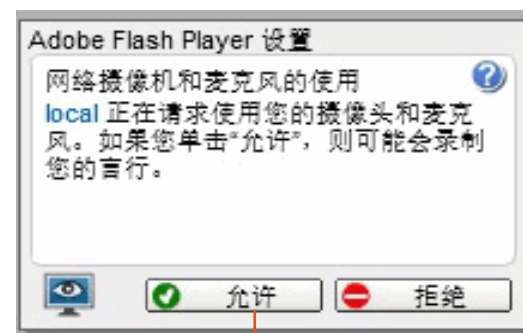
```
<html>
<head>
<meta http-equiv="content-type" content="text/html">
<title>A</title>
</head>
<body>
  <a href="B.html">
    This is B.
    
  </a>
</body>
</html>
```

禁用浮动



FLASH点击劫持

- 攻击者通过Flash构造点击劫持，在完成一系列复杂动作后，最终控制用户的摄像头。目前Adobe已修复此漏洞。



Real Cursor

Fake Cursor

<http://feross.org/webcam-spy/>

浏览器拖拽劫持

- ❑ 目前很多浏览器开始支持Drag&Drop的API。拖拽使得用户体验佳，操作方便。浏览器支持的拖拽对象有超链接、文字、窗口等。
- ❑ 浏览器拖拽不受同源策略限制。
- ❑ 拖拽劫持诱使用户从隐藏的iframe中拖拽出攻击者希望得到的数据，然后放到攻击者能够控制的另外一个页面中，从而窃取数据。
- ❑ 在JavaScript的支持下，拖拽劫持攻击更加隐蔽，能够突破传统ClickJacking的一些局限，造成更大的破坏。



浏览器拖拽劫持

- ❑ 国内的安全研究者xisigr 曾构造了一个针对窃取Gmail数据的拖拽劫持的网页小游戏，在小球和海豹的顶部都有隐藏的iframe。
- ❑ 当用户拖拽小球时，实际上选中了隐藏的iframe里的数据；在放下小球时，把数据也放在了隐藏的textarea中，从而完成一次数据窃取的过程。

隐藏的
IFrame

Gmail数据



触屏劫持

- ❑ 2010年9月，智能手机上的“触屏劫持”攻击被斯坦福的安全研究者公布，将其称为TapJacking。
- ❑ 以苹果公司的iPhone为代表，智能手机为人们提供了更先进的操控方式：触屏。从手机OS的角度来看，触屏实际上是一个事件，手机OS捕捉这些事件，并执行相应的动作，如：touchstart, touchend, touchmove, touchcancel。
- ❑ 手机屏幕范围有限，一些浏览器为了节约屏幕空间，甚至隐藏浏览器地址栏，导致手机上的视觉欺骗更加容易实施。



触屏劫持

- ❑ 智能手机用户以为自己在登录某个账号，实际上却按到了一个隐藏页面的按钮上，用户名和密码信息就这样被悄悄窃取。



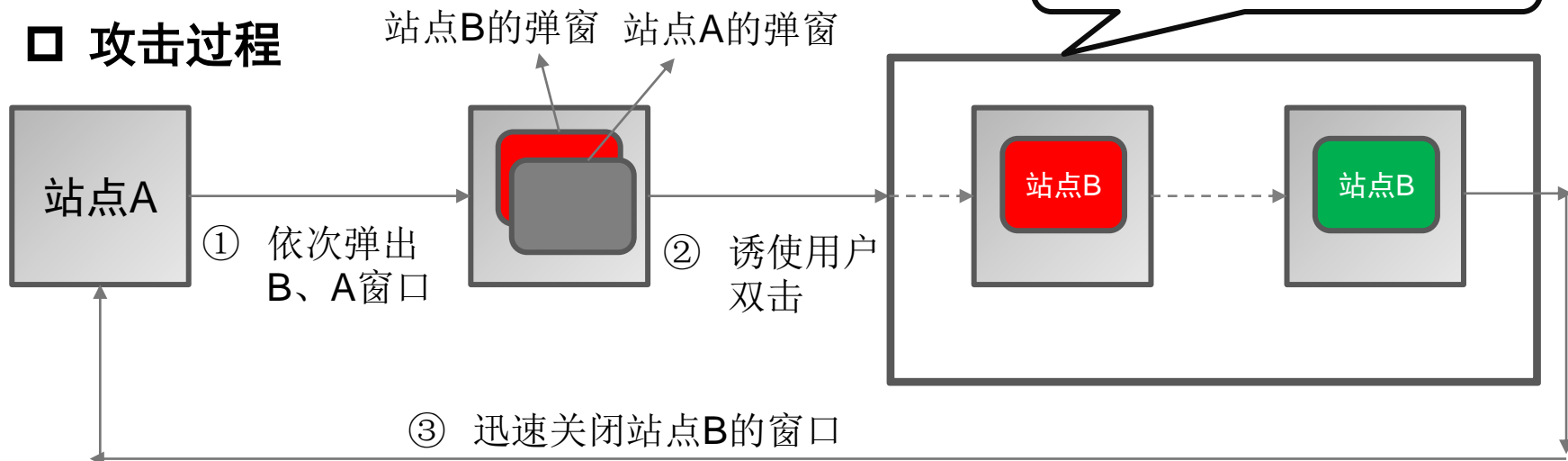
DOUBLE-CLICK TIMING ATTACK

□ 又称 “Cross Site Double Clicking Attack”

□ 攻击原理

此攻击主要利用人类对于快速的视觉变化难以作出反应，通过弹窗方式打破了framebusting的限制，诱使用户双击，导致用户在不知不觉中对另一个站点进行了操作。

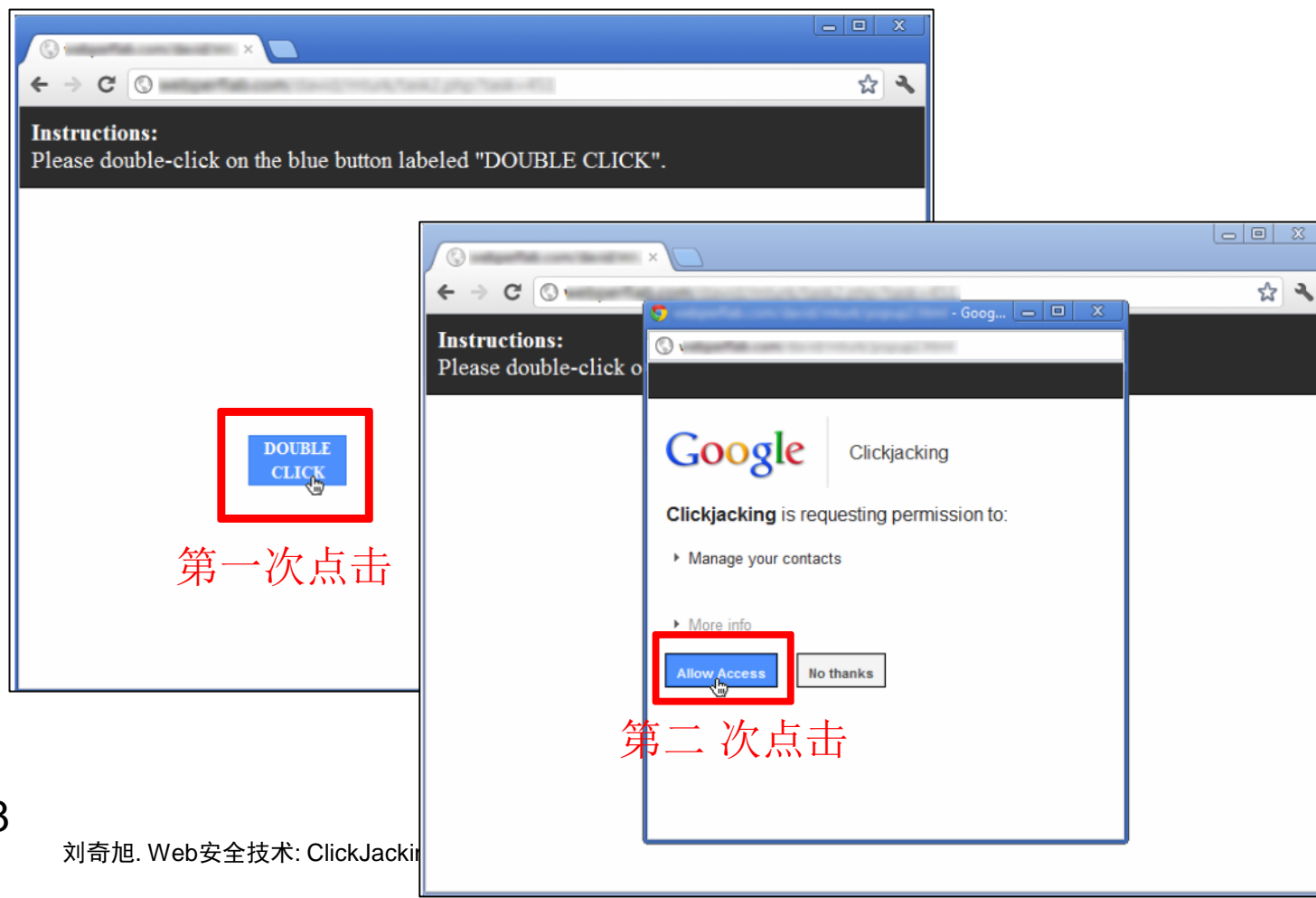
□ 攻击过程



DOUBLE-CLICK TIMING ATTACK

□ 谷歌OAuth授权案例

通过实施“双击”攻击，窃取用户的谷歌账户的私人数据。



Clickjacking危害仅限于此？



CLOAK & DAGGER ATTACK

2017年

IEEE S&P
2017

Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop

Yanick Fratantonio
UC Santa Barbara
yanick@cs.ucsb.edu

Chenxiong Qian, Simon P. Chung, Wenke Lee
Georgia Tech
qchenxiong3@gatech.edu
pchung34@mail.gatech.edu
wenke.lee@gmail.com

CVE-2017-
0752

Android Toast Overlay Attack: “Cloak and Dagger” with No Permissions



By Cong Zheng, Wenjun Hu, Xiao Zhang and Zhi Xu

September 7, 2017 at 1:00 PM

Category: Unit 42 Tags: Android, Cloak and Dagger



CLOAK & DAGGER ATTACK

□ 此攻击将允许攻击者悄悄拿到Android设备的更多的控制权

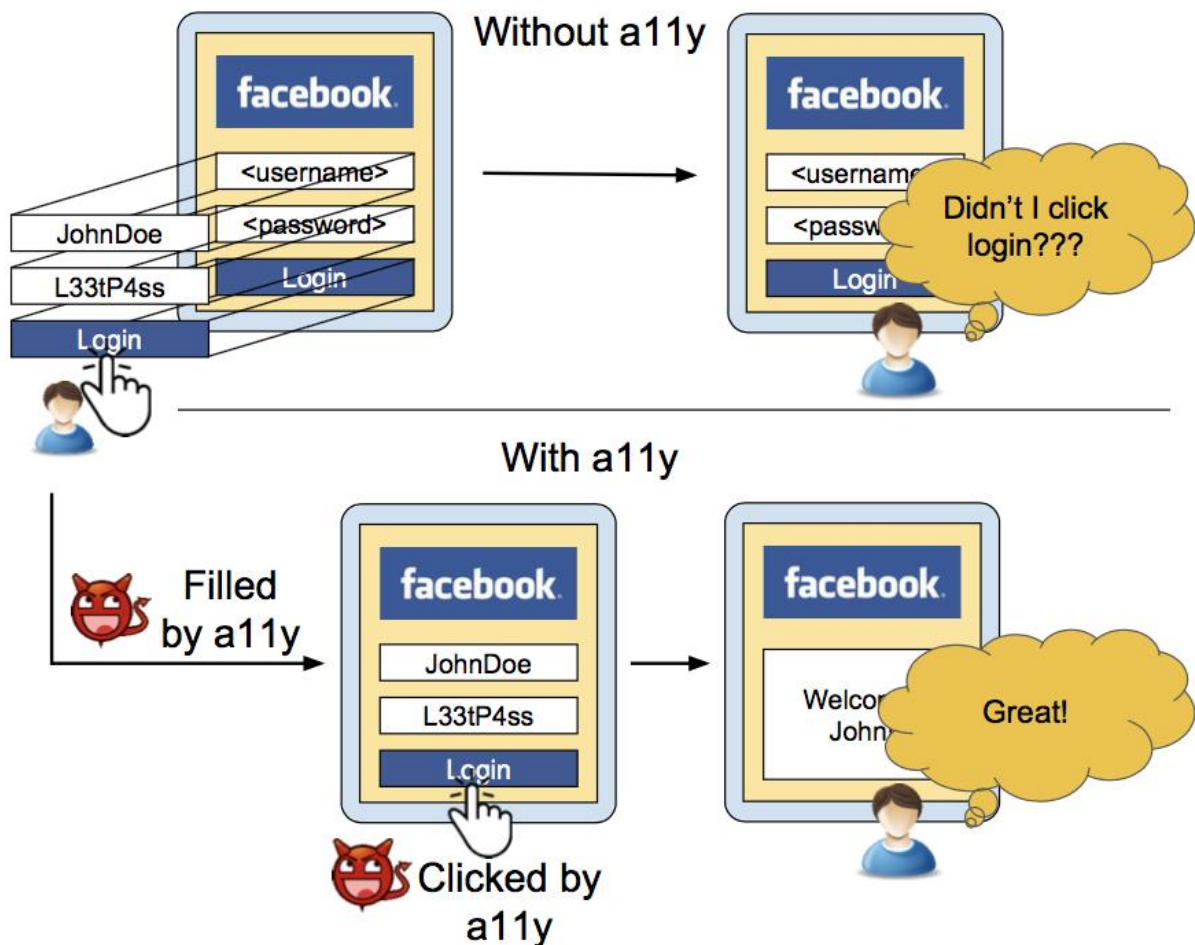
□ 利用Android设备中合法App的权限

- SYSTEM_ALERT_WINDOW (“draw on top”) 允许App在设备屏幕上显示额外的叠加窗口来覆盖其他的App界面。
- BIND_ACCESSIBILITY_SERVICE (“a11y”) 启动无障碍服务。帮助盲人和视力障碍用户在内的残疾人用户更好地通过语音命令来输入信息或通过屏幕阅读功能来了解屏幕内容。

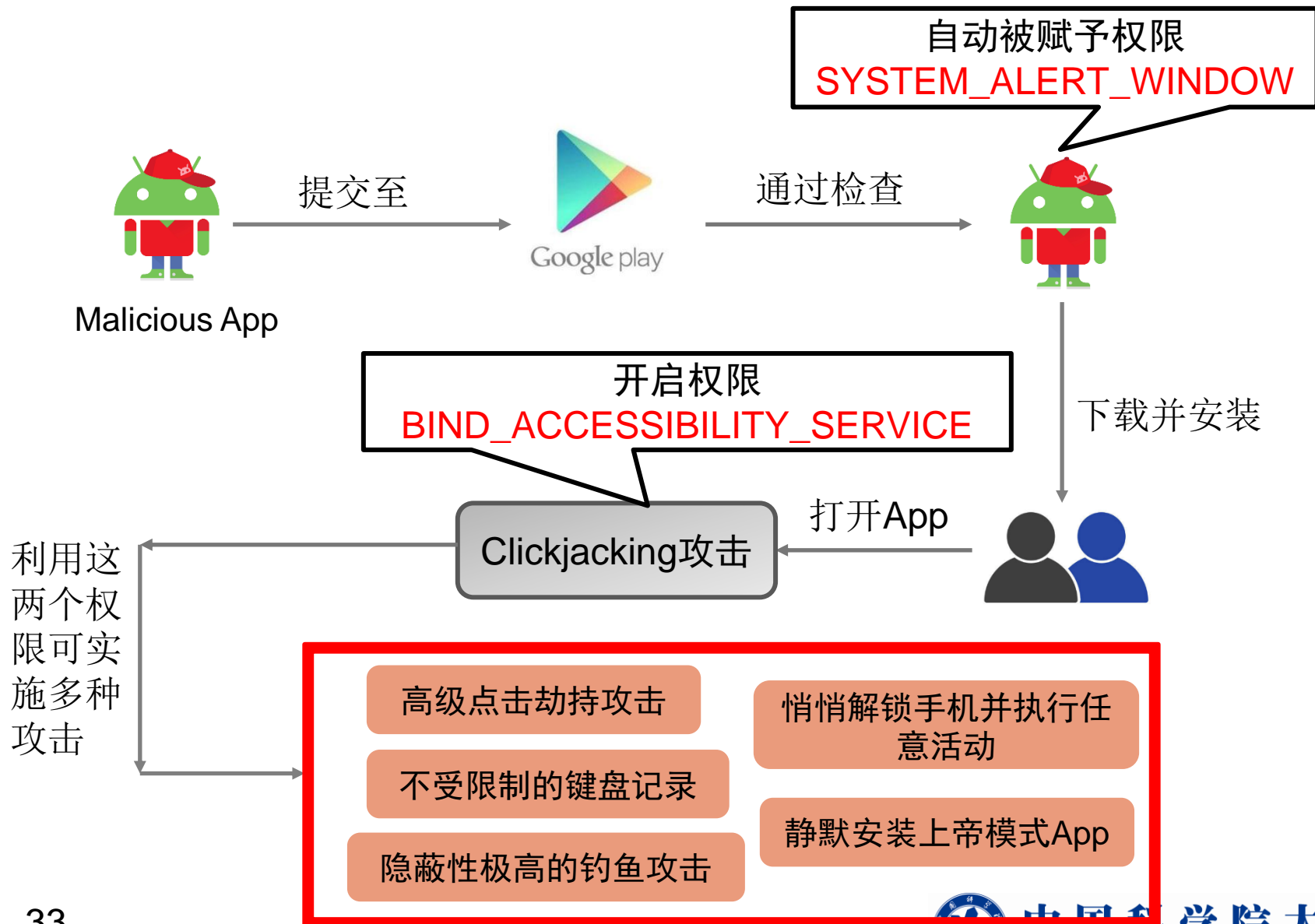


CLOAK & DAGGER ATTACK

□ 很多clickjacking攻击仅利用第一个权限，隐蔽性不高。



CLOAK & DAGGER ATTACK



Cloak & Dagger
Clickjacking to a11y
+
Silent God-mode app install

Cloak & Dagger Stealthy Phishing

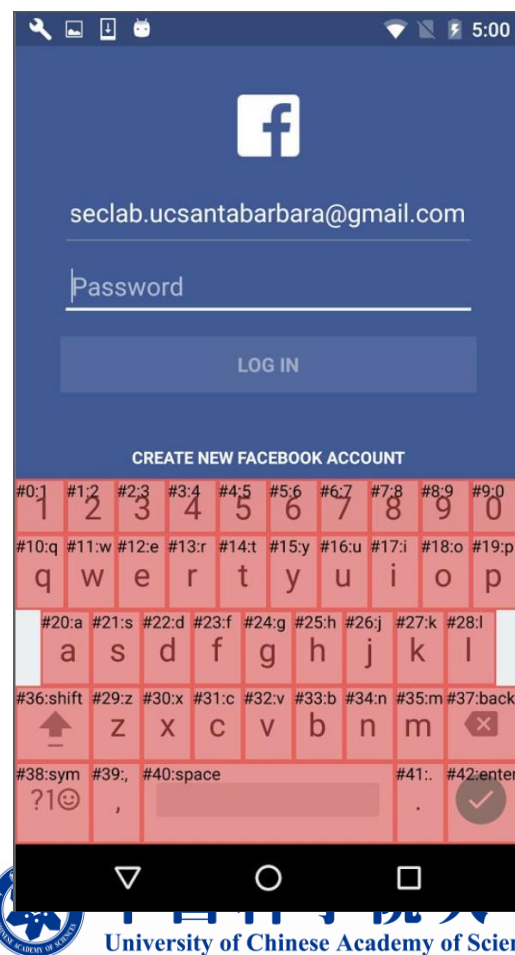
CLOAK & DAGGER ATTACK

□ Keystroke Inference

- 当页面中同时出现多个需要点击的地方，依照前述方法，很难记录所有点击的位置以及次序。

□ 解决方法

- 将所有的overlay依照一定的次序创建并以栈的方式组织起来。
- 利用obscured flag。当点击第*i*层overlay时，0~*i*-1层的overlay的obscured flag被设置为1。根据obscured flag被设置为1的个数判断哪个overlay被点击。



Cloak & Dagger Invisible Grid Attack (only uses SYSTEM_ALERT_WINDOW)

NEW CLOAK & DAGGER ATTACK

□ CVE-2017-0752

- Android Toast Overlay攻击
- Palo Alto Networks公司Unit 42实验室研究人员在Android overlay系统中发现一个高危漏洞，它允许使用“Toast类型”叠加层，可以发起新型的Android overlay攻击。
- 所有OS版本 < 8.0的Android设备都受到这个漏洞的影响
- 只需一个”BIND_ACCESSIBILITY_SERVICE”的权限



NEW TOAST OVERLAY ATTACK

□ 利用Toast进行新型攻击

- Toast窗口是Android上支持的overlay类型之一。
- 使用Toast窗口作为一个覆盖窗口，允许应用程序在另一个应用程序的界面上写入，而不需要特殊请求SYSTEM_ALERT_WINDOW权限。



TrendLabs  SECURITY
INTELLIGENCE Blog
SECURITY NEWS DIRECT FROM THREAT DEFENSE EXPERTS

Home Categories

Home » Exploits » Toast Overlay Weaponized to Install Several Android Malware

Toast Overlay Weaponized to Install Several Android Malware

Posted on: November 9, 2017 at 5:58 am Posted in: Exploits, Mobile, Vulnerabilities
Author: Lorin Wu (Mobile Threats Analyst)

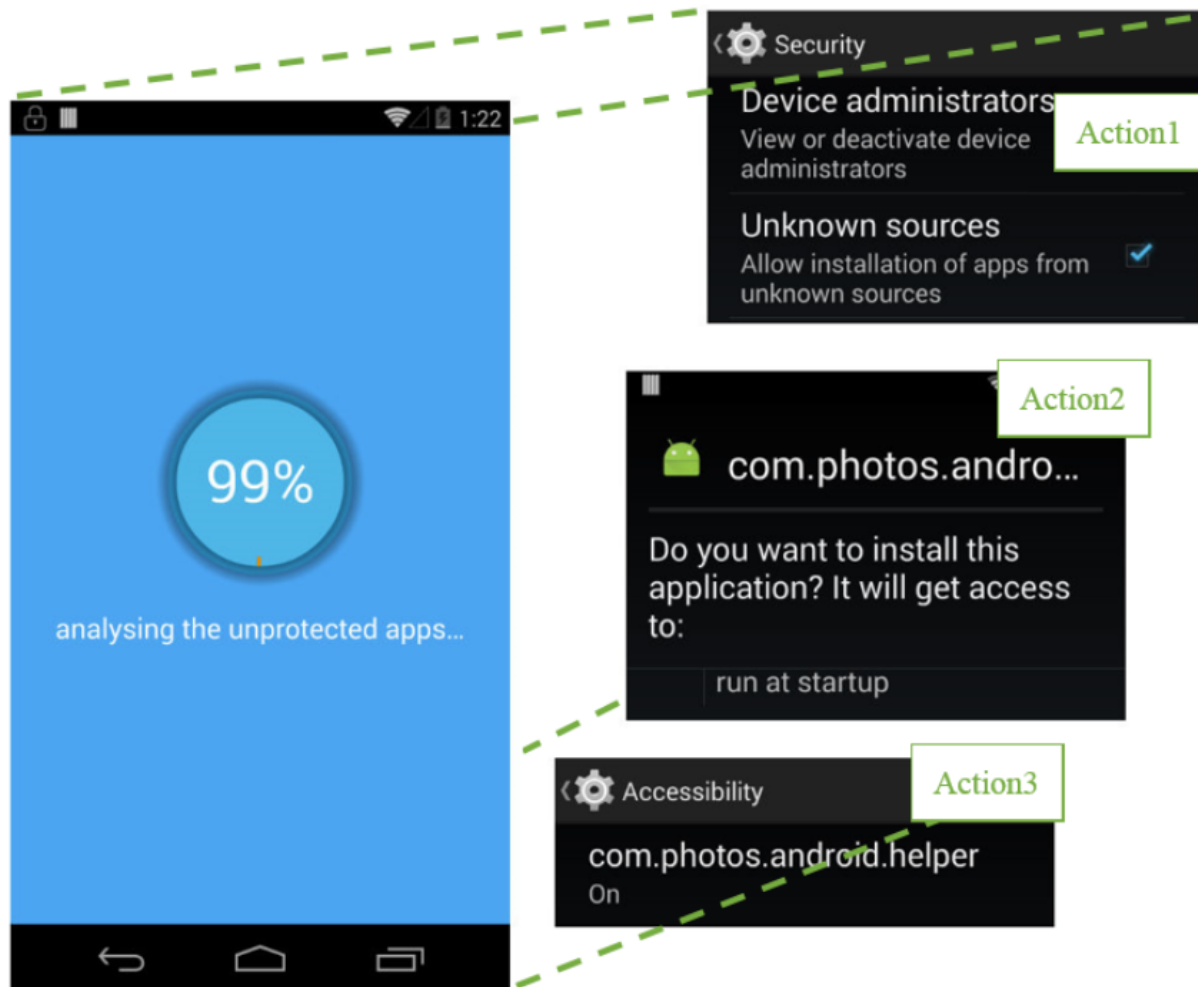
We uncovered new Android malware that can surreptitiously install other malware on the affected device via the **Toast Overlay attack**: TOASTAMIGO, detected by Trend Micro as ANDROIDOS_TOASTAMIGO. The malicious apps, one of which had installs between 100,000 and 500,000 as of November 6, 2017, abuses Android's Accessibility features, enabling them—at least for now—to have ad-clicking, app-installing and self-protecting/persistence capabilities.

Overlay attacks entail drawing and superimposing Android View (i.e., images, buttons) atop other running apps, windows or processes. A typical scenario for a Toast Overlay attack is to employ it to trick the user into clicking a window or button specified by the attacker instead of the legitimate one. The technique, which was demonstrated earlier this year, leverages a vulnerability in Toast (**CVE-2017-0752**, patched last September), a feature in Android used to display notifications over other applications.



TOAST OVERLAY攻击案例

- ❑ 截止2017年6月，其中一个恶意应用程序已经被下载安装了50多万次。
- ❑ 一个看似正常的图像(左)叠加在恶意软件触发的实际操作上，例如请求辅助功能。



A NEW ERA IN MOBILE BANKING TROJANS

Svpeng通过无障碍服务，打开键盘，窃取一切

2017年7月中旬，我们发现了知名手机银行恶意软件系列Svpeng的新修改-特洛伊木马-Banker.AndroidOS.Svpeng.ae。在这次修改中，网络罪犯增加了新的功能：它现在也可以作为键盘记录器，通过使用无障碍服务窃取输入的文本。

无障碍服务通常为残疾用户或那些暂时无法与设备完全交互的用户提供用户界面（UI）增强功能，可能是因为他们正在开车。滥用此系统功能会使特洛伊木马程序不仅可以从设备上安装的其他应用程序中窃取输入的文本，还可以授予自己更多的权限和权限，并阻止试图卸载特洛伊木马的行为。

攻击数据表明此特洛伊木马尚未广泛部署。在一周的时间里，我们观察到只有少数用户受到攻击，但这些目标跨越了23个国家。大多数被攻击的用户来自俄罗斯（29%）、德国（27%）、土耳其（15%）、波兰（6%）和法国（3%）。值得注意的是，尽管大多数受攻击的用户来自俄罗斯，但该特洛伊木马在运行俄语的设备上无法运行。这是俄罗斯网络犯罪分子逃避侦查和逮捕的标准策略。

Svpeng恶意软件家族以创新著称。从2013年开始，它是最早开始的[攻击短信银行](#)、[使用仿冒网页覆盖其他应用窃取凭据](#)、[以及封锁设备和要钱](#)。2016年，网络罪犯[积极分配Svpeng通过AdSense使用Chrome浏览器中的漏洞](#)。这使得Svpeng成为最危险的移动恶意软件家族之一，这也是我们监视新版本功能的原因。

<https://securelist.com/a-new-era-in-mobile-banking-trojans/79198/>



A NEW ERA IN MOBILE BANKING TROJANS

攻击过程

启动后，特洛伊木马-Banker.AndroidOS.Svpeng.ae会检查设备语言，如果不是俄语，则会请求设备允许使用辅助功能服务。在滥用这一特权时，它可以做许多有害的事情。它授予自己设备管理员权限，使自己超越其他应用程序，将自己安装为默认SMS应用程序，并授予自己一些动态权限，包括发送和接收SMS、拨打电话和读取联系人的能力。此外，利用其新获得的功能，特洛伊木马可以阻止任何试图删除设备管理员权限的行为，从而阻止其卸载。有趣的是，在这样做的时候，它也阻止了为任何其他应用程序添加或删除设备管理员权限的任何尝试。

使用辅助功能服务，特洛伊木马程序可以访问其他应用程序的UI，并从中窃取数据，如接口元素的名称及其内容（如果可用）。这包括输入的文本。此外，每次用户按下键盘上的按钮时，它都会截图，并将它们上载到恶意服务器。它不仅支持标准的Android键盘，还支持一些第三方键盘。

一些应用程序，主要是银行应用程序，不允许在顶部截图。在这种情况下，特洛伊木马还有另一种窃取数据的选择，即在受攻击的应用程序上设置钓鱼窗口。有趣的是，为了找出哪个应用程序在上面，它还使用了辅助功能服务。

从Svpeng从其命令控制服务器（CnC）接收到的信息，我能够截获一个加密的配置文件并对其进行解密，以找出受攻击的应用程序，并获得带有钓鱼网页的URL。

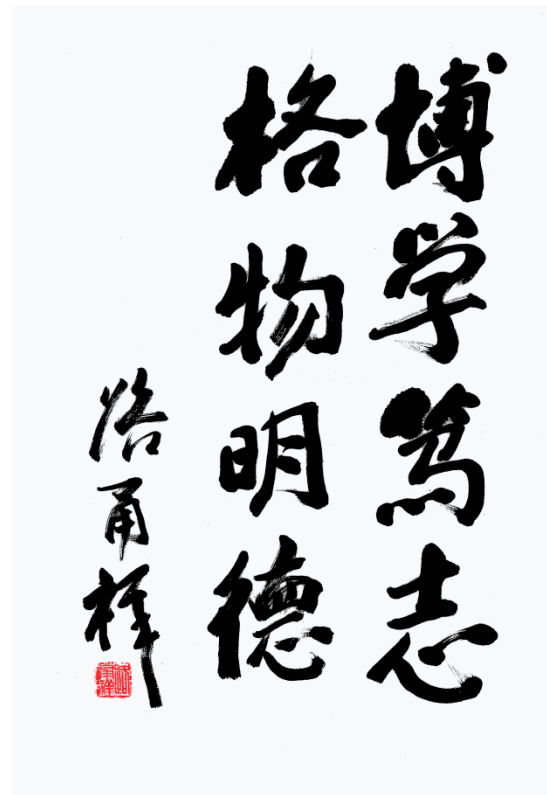
我发现了一些特洛伊木马试图阻止的防病毒应用程序，以及一些带有钓鱼网址的应用程序。像大多数手机银行家一样，Svpeng会覆盖一些谷歌应用程序来窃取信用卡的详细信息。

<https://securelist.com/a-new-era-in-mobile-banking-trojans/79198/>



本章大纲

- 概述
- 攻击类型
- 防御方法



XFO

❑ X-FRAME-OPTIONS (XFO)是微软提出的一个http头，用来防御利用iframe嵌套的点击劫持攻击。在IE8、Firefox3.6、Chrome4以上的版本均能很好的支持。

❑ X-FRAME-OPTIONS属性值

DENY // 拒绝任何域加载

SAMEORIGIN // 允许同源域加载到iframe或frame

ALLOW-FROM **URI** // 允许指定的Uri嵌入到iframe或frame



□ APACHE配置X-FRAME-OPTIONS

- 在站点配置文件httpd.conf中添加如下配置，限制只有站点内的页面才可以嵌入iframe。

Header always append X-Frame-Options SAMEORIGIN

- 如果同一apache服务器上有多多个站点，只想针对一个站点进行配置，可以修改.htaccess文件，添加如下内容：

Header append X-FRAME-OPTIONS "SAMEORIGIN"

XFO

❑ NGINX 配置X-FRAME-OPTIONS

- ❑ 到 nginx/conf 文件夹下，修改nginx.conf，添加如下内容并重启：

`add_header X-Frame-Options "SAMEORIGIN";`

```
server {  
    listen      80;  
    server_name localhost;  
    server_tokens off;  
    add_header X-Frame-Options "SAMEORIGIN";  
}
```



XFO

□ IIS配置X-FRAME-OPTIONS

□ 在web站点的web.config中配置如下：

```
<add name="X-Frame-Options" value="SAMEORIGIN" />
```

```
<system.webServer>  
...  
<httpProtocol>  
  <customHeaders>  
    <add name="X-Frame-Options" value="SAMEORIGIN" />  
  </customHeaders>  
</httpProtocol>  
...  
</system.webServer>
```



FRAME BUSTING

❑ 针对传统的点击劫持，一般是通过禁止跨域的iframe来防范；通常可以写一段**JavaScript**代码，以禁止iframe的嵌套。这种方法叫做Frame Busting。

❑ 例如下列代码：

```
if(top.location !=location) {  
    top.location=self.location;  
}
```



FRAME BUSTING

□ 常见的Frame Busting有以下方式:

```
if (top !== self)
if (top.location !== self.location)
if (top.location !== location)
if (parent.frames.length > 0)
if (window !== top)
if (window.top !== window.self)
if (window.self !== window.top)
if (parent && parent !== window)
if (parent && parent.frames && parent.frames.length>0)
if((self.parent&&!(self.parent===self))&&(self.parent.frames.length!=0))
top.location = self.location
top.location.href = document.location.href
top.location.href = self.location.href
top.location.replace(self.location)
top.location.href = window.location.href
```



FRAME BUSTING (续)

□ 常见的Frame Busting有以下方式:

```
top.location.href = window.location.href
top.location.href = "URL"
document.write('')
top.location = location
top.location.replace(document.location)
top.location.replace('URL')
top.location.href = document.location
top.location.replace(window.location.href)
top.location.href = location.href
self.parent.location = document.location
parent.location.href = self.document.location
top.location.href = self.location
top.location = window.location
top.location.replace(window.location.pathname)
windowwindow.top.location = window.self.location
setTimeout(function(){document.body.innerHTML='';},1);
```



CSP: frame-ancestors

跳转到: [Syntax](#) [Examples](#) [Specifications](#) [Browser compatibility](#) [See also](#)

Web 开发技术 > HTTP > HTTP Headers >

Content-Security-Policy >

CSP: frame-ancestors

相关主题

[HTTP](#)

Guides:

- ▶ [Resources and URIs](#)
- ▶ [HTTP guide](#)
- ▶ [HTTP security](#)

[HTTP access control \(CORS\)](#)

[HTTP authentication](#)

HTTP头部 **Content-Security-Policy** (CSP) 中的 **frame-ancestors** 指令指定了一个可以包含 `<frame>`, `<iframe>`, `<object>`, `<embed>`, or `<applet>` 等元素的有效父级。

当该指令设置为 'none' 时, 其作用类似于 **X-Frame-Options : DENY** (该头部被一些老版本浏览器所支持)。

CSP版本 (CSP version)	2
指令类型 (Directive type)	Navigation directive
是否后备使用 <code>default-src</code>	否。如未设置则允许所有可能值。
该指令不支持通过 <code><meta></code> 元素或通过 Content-Security-policy-Report-Only 头域所指定。	

iframe不起作用？你可能碰到它了。

有一个需求要在iframe里显示一个网站，但设置iframe的src后，iframe并没有起作用。然后打开控制台，发现错误如下：

```
✖ Refused to display 'https://tours.wingontravel.com/' in a frame because an ancestor violates the following Content Security Policy directive: "frame-ancestors 'self'".
```

对其搜索找到了答案：<https://stackoverflow.com/questions/37799258/content-security-policy-directive-frame-ancestors-self>

该网站设置了响应头frame-ancestors 'self'，

▼ Response Headers [view source](#)

```
Cache-Control: no-cache, no-store, must-revalidate
Content-Encoding: gzip
CONTENT-SECURITY-POLICY: frame-ancestors 'self'
Content-Type: text/html; charset=utf-8
Date: Wed, 02 May 2018 06:43:28 GMT
Expires: -1
Pragma: no-cache
Server: Microsoft-IIS/7.5
Strict-Transport-Security: max-age=31536000; includeSubDomains
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-AspNet-Version: 4.0.30319
X-AspNetMvc-Version: 5.2
X-CONTENT-TYPE-OPTIONS: nosniff
X-FRAME-OPTIONS: SAMEORIGIN
X-Powered-By: ASP.NET
X-XSS-PROTECTION: 1; mode=block
```

从而禁止iframe嵌入自己的网站。

<https://www.cnblogs.com/luanfujian/archive/2018/05/02/8980521.html>

XFO VS. CSP

XFO vs. CSP frame-ancestors

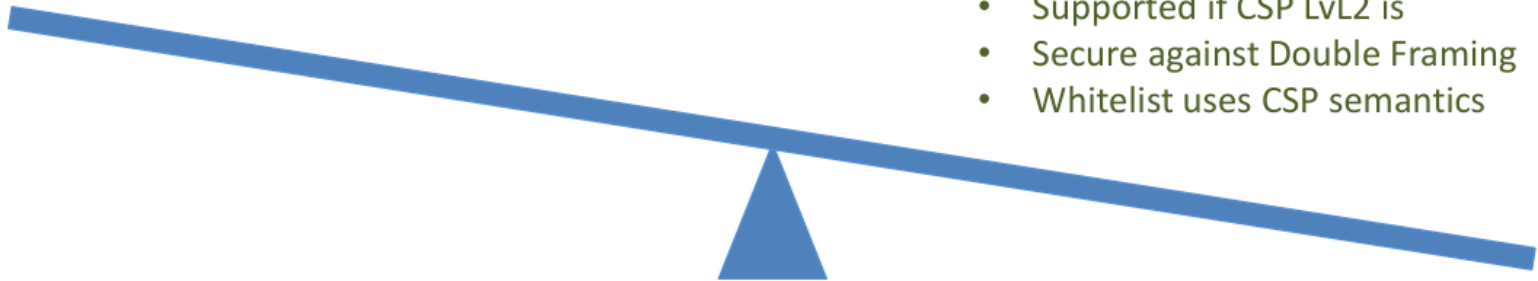


X-Frame-Options:

- Deprecated since 2012
- Inconsistently implemented
- Only Partially supported
- Double Framing attacks
- Only one whitelisted entry

CSP frame-ancestors:

- Well defined standard
- Supported if CSP LvL2 is
- Secure against Double Framing
- Whitelist uses CSP semantics



<https://apapedulimu.click/clickjacking-on-google-myaccount-worth-7500/>

可以绕过？

Clickjacking on Google MyAccount Worth 7,500\$

Recently, I got surprised from google, I found bug Clickjacking On Google My account. And they reward me 7,500\$ for single bug. Amazing, right?. This bug I've found on March 2018, but the clickjacking is just blocked by CSP, and on August, I've found way to bypass it.

GOOGLE MYACCOUNT 点击劫持案例

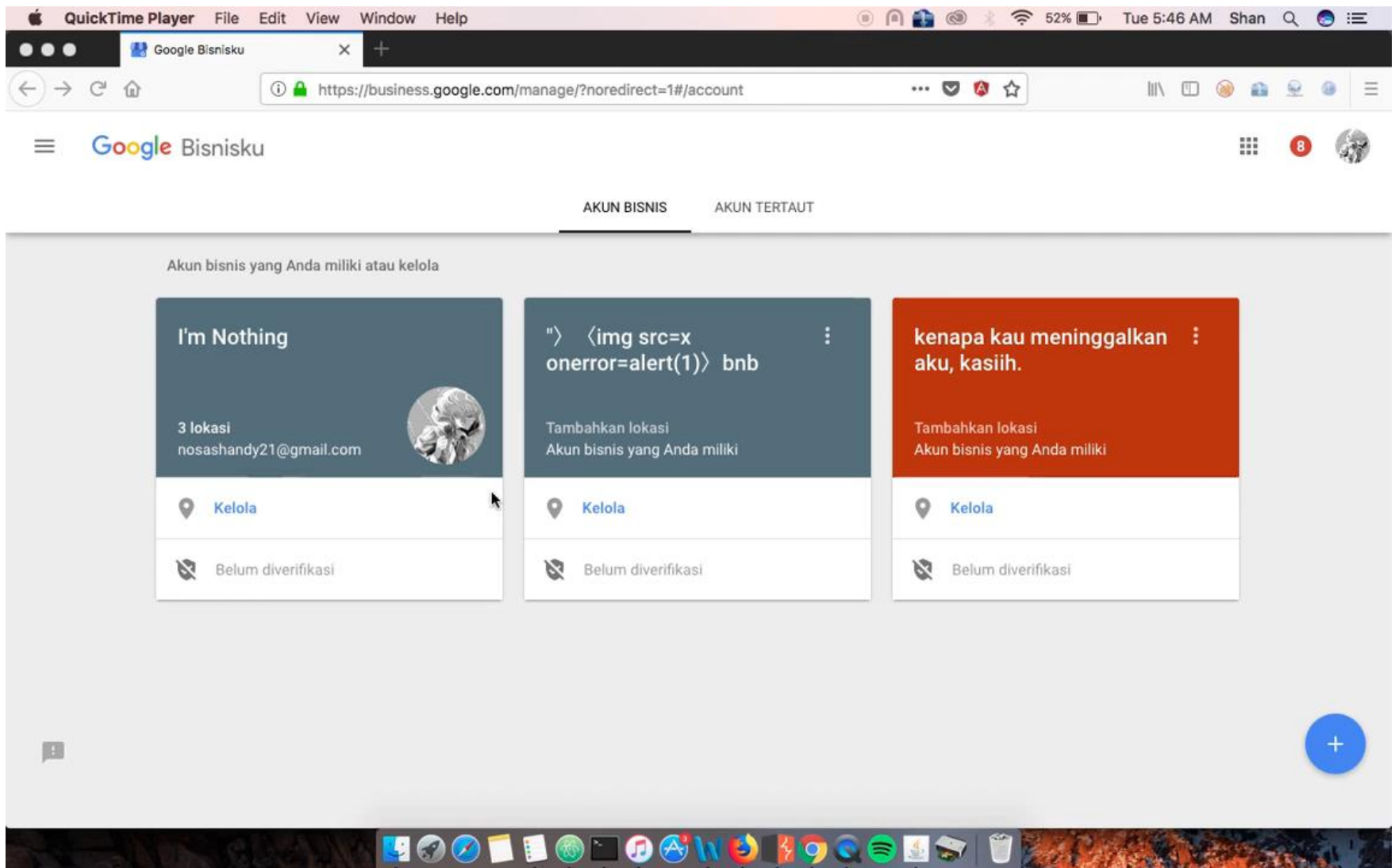
- ❑ 利用放入URL编码绕过CSP，成功实施clickjacking。
- ❑ 使用 **origin=https://business.google.com** 可以绕过CSP，但是此子域名却不被允许使用。
- ❑ 通过将 **origin=https://%0d.business.google.com**，可以成功绕过CSP。

```
<iframe  
src="  
https://myaccount.google.com/u/0/brandaccounts/group/{your-  
group-id}/managers?  
originProduct=AC&origin=https://%0d.business.google.com"  
width="1000" height="1000">
```

<https://apapedulimu.click/clickjacking-on-google-myaccount-worth-7500/>



CLICKJACKING BYPASS CSP





安装浏览器插件


- ❑ NoScript是一个免费和开源的，为Mozilla Firefox和Mozilla Application Suite网页浏览器（诸如Flock、SeaMonkey等）所开发的扩展（Add-ons）。
- ❑ NoScript允许JavaScript, Java, Flash, Sliverlight以及其它插件和脚本内容基于白名单被选择性的执行。
- ❑ NoScript在很大程度上能增强浏览器的安全性，对XSS, CSRF和Clickjacking可以起到防护作用。

选项 (Q)...


- ✓ 显示被阻止脚本的信息
- ✓ 在浏览器底部显示信息
- 脚本被阻止时提示声音

 全局允许 JavaScript (危险)


 *Temporarily allow all this page*


 不可信任

 允许 feedsky.com


 临时允许 feedsky.com

 禁止 googlesyndication.com

 允许 baidu.com

 临时允许 baidu.com

 允许 appinn.com

 临时允许 appinn.com



HTML5防御

- 在HTML 5中，专门为iframe定义了一个新的属性，叫做sandbox；使用sandbox这一个属性后，<iframe>标签加载的内容将被视为一个独立的“源”，其中的脚本将被禁止执行，表单被禁止提交，插件被禁止加载，指向其他浏览器对象的连接也会被禁止。



HTML5防御

语法

```
<iframe sandbox="value">
```

属性值

值	描述
""	应用以下所有的限制。
allow-same-origin	允许 <code>iframe</code> 内容被视为与包含文档有相同的来源。
allow-top-navigation	允许 <code>iframe</code> 内容从包含文档导航（加载）内容。
allow-forms	允许表单提交。
allow-scripts	允许脚本执行。





A Tale of Two Headers: A Formal Analysis of Inconsistent Click-Jacking Protection on the Web

Stefano Calzavara, *Università Ca' Foscari Venezia*; Sebastian Roth, *CISPA Helmholtz Center for Information Security and Saarbrücken Graduate School of Computer Science*; Alvisè Rabitti, *Università Ca' Foscari Venezia*; Michael Backes and Ben Stock, *CISPA Helmholtz Center for Information Security*

<https://www.usenix.org/conference/usenixsecurity20/presentation/calzavara>

**This paper is included in the Proceedings of the
29th USENIX Security Symposium.**

August 12-14, 2020

978-1-939133-17-5

Research Questions

- Can we formally describe the inconsistency between the XFO header and CSP frame-ancestors?
- How inconsistent is framing control implemented in different browsers / deployed on real-world Web sites?
- Can we automatically fix inconsistencies?

Browser Semantics for Framing Control

Browser	CSP	ALLOW-FROM	Multiple Headers	Header Parsing	Double Framing
Chrome	✓	✗	✓	✓	✓
Chrome (Android)	✓	✗	✓	✓	✓
Edge	✓	✓	✗	✗	✗
Firefox	✓	✓	✓	✓	✓
Internet Explorer	✗	✓	✗	✗	✗
Opera Mini	✗	✗	✗	✗	✓
Safari	✓	✗	✓	✓	✓
Safari (iOS)	✓	✗	✓	✓	✓
Samsung Internet	✓	✗	✓	✓	✓
UC Browser	✓	✗	✓	✓	✗

本章小结

- ❑ ClickJacking攻击形式包含图片覆盖劫持、Flash点击劫持、浏览器拖拽劫持、触屏劫持。
- ❑ X-FRAME-OPTIONS和Frame Busting是应对ClickJacking攻击的常用防御方式，在Apache、Nginx、IIS web 服务器上均可配置X-Frame-Options。
- ❑ HTML 5中加入了一些新的安全元素属性，如：sandbox，可以防止ClickJacking带来的恶意攻击。



延伸阅读

USENIX
Security 2012

Clickjacking Revisited A Perceptual View of UI Security

*Devdatta Akhawe, Warren He, Zhiwei Li, Reza Moazzezi, Dawn Song
UC Berkeley*

Clickjacking: Attacks and Defenses

Lin-Shung Huang
Carnegie Mellon University
linshung.huang@sv.cmu.edu

Alex Moshchuk
Microsoft Research
alexmos@microsoft.com

Helen J. Wang
Microsoft Research
helenw@microsoft.com

Stuart Schechter
Microsoft Research
stuart.schechter@microsoft.com

Collin Jackson
Carnegie Mellon University
collin.jackson@sv.cmu.edu

Busting Frame Busting: a Study of Clickjacking Vulnerabilities on Popular Sites

Gustav Rydstedt, Elie Bursztein, Dan Boneh
Stanford University
{rydstedt, elie, dabo}@stanford.edu

Collin Jackson
Carnegie Mellon University
collin.jackson@sv.cmu.edu

June 7, 2010

Clickjacking Defense Cheat Sheet



USENIX
Security 2020

A Tale of Two Headers: A Formal Analysis of Inconsistent Click-Jacking Protection on the Web

Stefano Calzavara, *Università Ca' Foscari Venezia*; Sebastian Roth, *CISPA Helmholtz Center for Information Security and Saarbrücken Graduate School of Computer Science*; Alvis Rabitti, *Università Ca' Foscari Venezia*; Michael Backes and Ben Stock, *CISPA Helmholtz Center for Information Security*

<https://www.usenix.org/conference/usenixsecurity20/presentation/calzavara>



延伸阅读

O'REILLY®

Web Application Security

Exploitation and Countermeasures
for Modern Web Applications



Andrew Hoffman

Web Application Security

by Andrew Hoffman

Copyright © 2020 Andrew Hoffman. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://oreilly.com>). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com.

Acquisitions Editor: Jennifer Pollock

Development Editor: Angela Rufino

Production Editor: Katherine Tozer

Copyeditor: Sonia Saruba

Proofreader: Christina Edwards, Piper Editorial

Indexer: Judy McConville

Interior Designer: David Futato

Cover Designer: Karen Montgomery

Illustrator: Rebecca Demarest

March 2020: First Edition

Revision History for the First Release

2020-03-03: First Release

2020-04-10: Second Release

2020-06-05: Third Release

See <http://oreilly.com/catalog/errata.csp?isbn=9781492053118> for release details.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Web Application Security*, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

The views expressed in this work are those of the author, and do not represent the publisher's views. While the publisher and the author have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the author disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

This work is part of a collaboration between O'Reilly and F5/NGINX. See our [statement of editorial independence](#).

978-1-492-08796-0

[LSI]



后续课程内容

- 第二部分：Web客户端安全
- 详细讲解XSS跨站、跨站点请求伪造、点击劫持等前端安全。
- 2.1 OWASP Top Ten
- 2.2 XSS与CSRF
- 2.3 ClickJacking
- 2.4 浏览器与扩展安全
- 2.5 案例分析



谢谢大家

刘奇旭

liuqixu@iie.ac.cn

中科院信工所 第六研究室



中国科学院大学
University of Chinese Academy of Sciences