中国人民大学 信息学院
RENMIN UNIVERSITY OF CHINA SCHOOL OF INFORMATION

12. 浏览器安全——核心安全机制及代码分析

授课教师：游伟 副教授

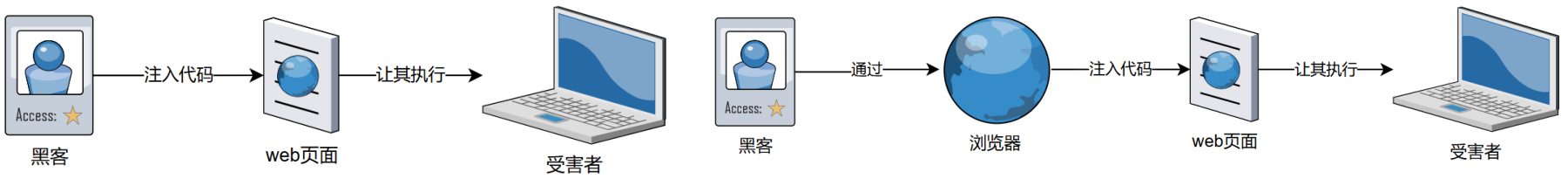
授课时间：周四18:00 – 19:30（立德楼513）

课程主页：https://www.youwei.site/course/websecurity

目录

12.1 同源策略概念及实现

- 我们在前端漏洞中提到跨站脚本攻击（XSS），XSS存在的条件为受害网站的代码存在漏洞，导致攻击者的脚本得以在受害网站上执行
- 然而还存在另一种危害更大的攻击方式——通用跨站脚本攻击（UXSS），顾名思义，它与网站的代码是否安全无关，只要存在该漏洞，任何网站都能受到威胁。
- UXSS是浏览器实现的漏洞。

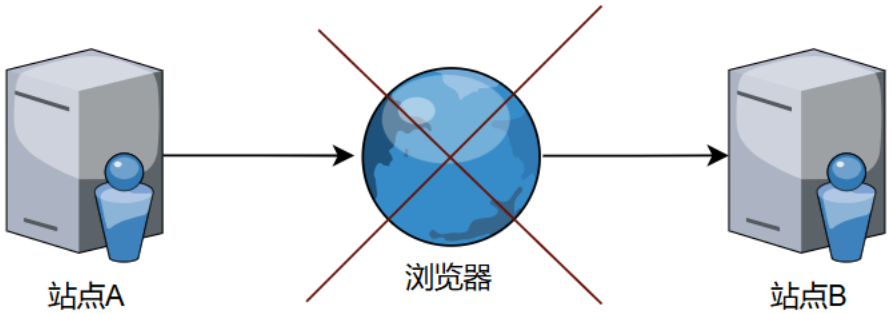


XSS　　UXSS

12.1 同源策略概念及实现

- 在起那面的攻击概念图中我们可以看到，不管是XSS还是UXSS，都是通过**将代码注入到受害网站**达成攻击目的，为什么要这么做？这就不得不提浏览器的一个重要安全措施：同源策略（SOP）。

- 同源策略被用于限制一个源的文档或者**它加载的脚本**如何能与另一个源的资源进行交互。同源策略能有效地阻止来自恶意网站的恶意代码、恶意文档对浏览器用户造成的不利影响，比如阻止攻击者获取用户存在浏览器中的敏感网站（比如银行）的登录信息（账号密码，cookie等）。

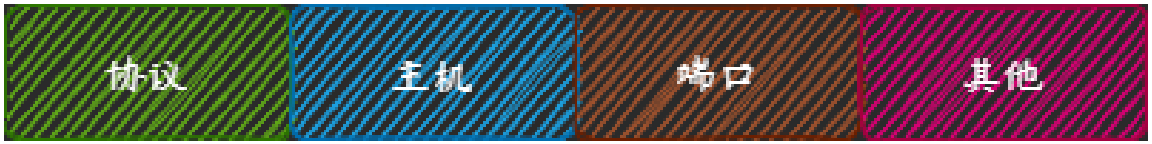


12.1.1 源

- 简单来说，同源策略控制着不同源之间的交互，并在一般情况下阻止跨源的**读取操作**。那么什么是源呢?
- 一个“源”包含了三个部分，分别是协议、主机（也被称为域名）、端口。我们通常在浏览器中访问的url如下：
- https://www.google.com:80/...
- 其中，https://，http://等被称为协议，www.google.com被称为主机，80被称为端口（由于Web服务的默认端口为80，所以80端口通常被省略，如果某个服务器的Web服务端口不是80，则不能省略），跟在端口后面的“/”之后的部分并不影响源，我们权且称之为“其他”。

12.1.1 源

- 更直观的，我们可以将一个url解析为下图

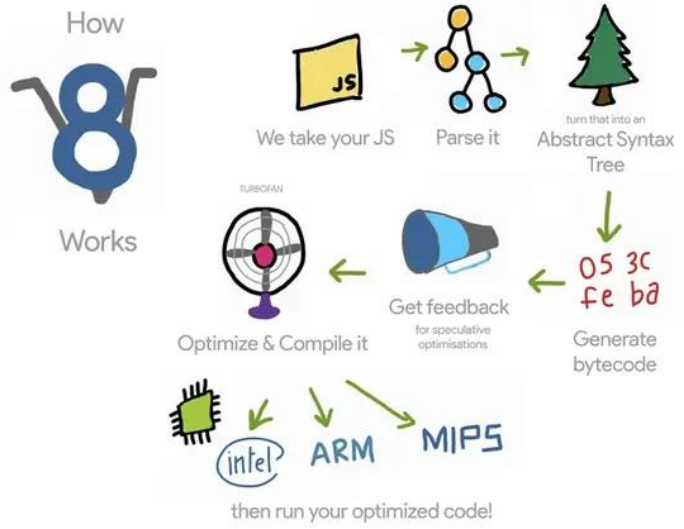


- 只要**前三个**属性有任何差别，我们都称为源是不同的，否则，源是相同的。下图列举了一些与https://www.google.com/hello_world.html相关的源。

https://	www.google.com	(:80)被省略	hello _ world.html	最初的源
https://	www.google.com	:80	hello _ world.html	同源，因为都是80端口
https://	www.google.com	(:80)被省略	print.html	同源，因为其他部分不影响
http://	www.google.com	(:80)被省略	hello _ world.html	不同源，协议不同
https://	www.baidu.com	(:80)被省略	hello _ world.html	不同源，主机不同
https://	www.google.com	:81	hello _ world.html	不同源，端口不同

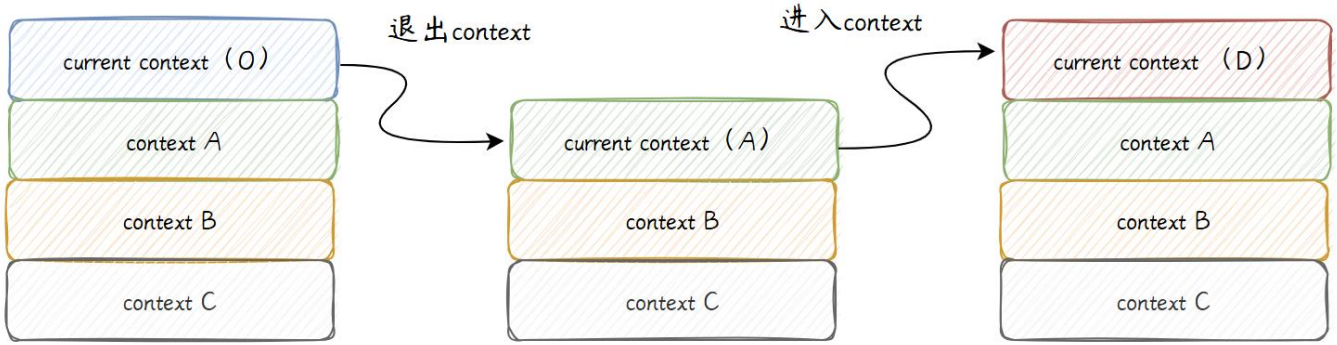
12.1.2 V8引擎

- Javascript已经成为当前Web服务前端的主要语言，当攻击者试图通过Web服务获取受害者的敏感信息时，使用Javascript是不可避免的。怎么保证来自不同源的文档和代码不能访问彼此？这是通过控制Javascript引擎的行为实现的。
- Chrome使用的Javascript引擎为V8，V8是一个高效的Javascript和WebAssembly开源引擎，它能使任何C++程序能够向Javascript代码公开自己的对象和函数，并且具有优秀的垃圾回收机制和安全模型。



上下文

- 基于同源策略的考虑，我们通常希望来自不同源的代码是相互隔离的，V8提供了一个类名为context（上下文），每个context都作为一个Javascript代码的执行环境并且相互隔离。
- 在Javascript层面，不同context是相互隔离的，也就是说，在一般情况下，contextA的JS代码是不能访问contextB的JS对象或函数的。
- 在C++层面，一旦一个context被创建，我们就可以随意的进出它，并且是一个类似于栈的方式，当我们从contextA进入contextB，再从contextB退出时，我们会回到contextA。

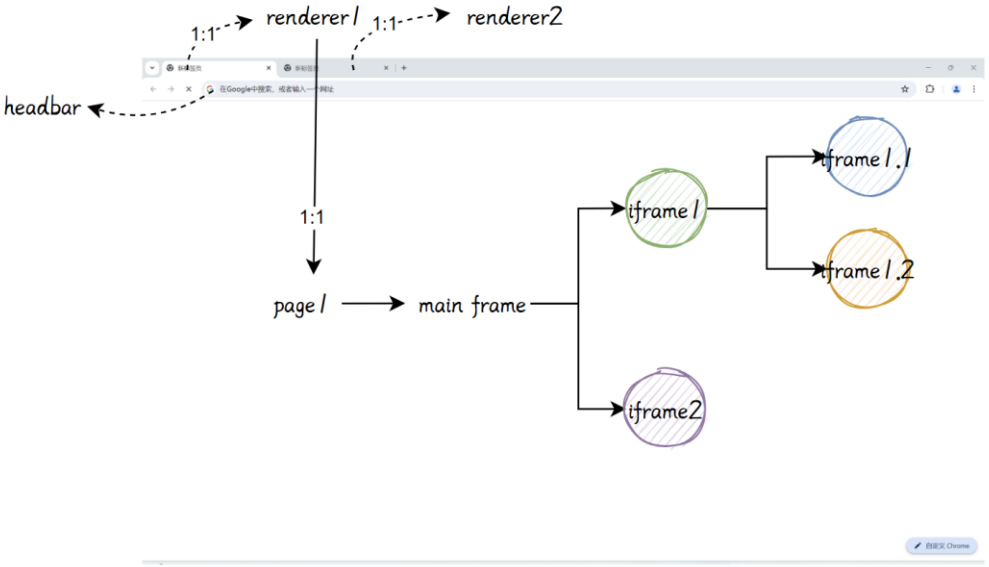


安全模型

- 在chrome中，每个源都对应着不同的context，更进一步的，每个window和iframe都对应着不同的context，这意味着它们都有着**崭新的Javascript环境**，而访问除了当前context之外的任何context都是默认禁止的，而想要在当前context访问其他context，就需要验证security token或者使用回调函数。
- 更进一步地，chrome引进了site isolation机制，将每个不同的源都放到了不同的进程中，由于进程之间有天然的隔离，使得每个源的信息更加难以被其他源获取。
- 了解更多：Chromium Docs - Process Model and Site Isolation

12.1.3 Chrome页面架构

- 一个浏览器大致可以分为下面三级：browser（浏览器），renderer（渲染器），frame（帧）
- 当我们打开一个浏览器时，整个浏览器就是一个browser进程，browser进程是所有其他进程的母进程，下图模拟了一个浏览器在视觉上的架构。



12.1.3 Chrome页面架构

- 每一个标签页都对应着一个独特的渲染器进程，而每一个renderer都对应着一个独特的页（page）
- page主要是用于管理一个renderer进程的文档，一个文档通常被称之为一个帧（frame）
- 当前页面的文档被称为main frame，以iframe形式插入到页面中的其他文档被称为local frame，所有的frame都由page管理。

```
class Page {
...
Member<Frame> main_frame_;
}
```

12.1.4 同源策略的实施

- 在安全模型中我们提到，同源策略的检验主要使用的就是上下文（context），这一部分主要讲的就是在chromium中上下文是如何设置的以及在使用JS代码访问某个对象或属性时chromium是如何利用context验证权限的。

DomWindow和ExcutionContext

- 通过chrome页面架构我们可以看到，当前页面所有需要显示或者加载的东西来自所有的frame（1个main frame + N个LocalFrame），chromium创建了一个C++类——DomWindow，用来对应Javascript中的窗口对象(window)。
- DomWindow包含了一个窗口的所有属性，并继承自一个关键类ExcutionContext，顾名思义，ExcutionContext是一个脚本运行的环境，它提供在Web上脚本运行的常见属性，他与我们之前提到的上下文（v8:context）的区别在于，它不是某个JS代码运行的环境，而是整个DomWindow运行的环境。

DomWindow和ExcutionContext

- 在chromium中，通常将DomWindow当作是ExcutionContext的具体实现，由于DomWindow与frame相绑定，它在确定唯一性方面比ExcutionContext更具优越性，所以大部分需要传递ExcutionContext的地方传递的实际上是与frame绑定的DomWindow。

```
class DomWindow : public ExcutionContext  {
...
Member<Frame> frame _ ;
}
```

SecurityOrigin和SecurityToken

- DomWindow包含了一个窗口的所有属性，与同源策略最相关的就是“源”这个属性了。在chromium中，为源定义了一个特殊标识，用来判断它是否能被其他源访问，这个特殊标识就是SecurityOrigin。
- 这个类中不仅以字符串形式存储着这个DomWindow的源，还给该源设置了一系列属性，其中就包括两个布尔变量：universal_access和cross_agent_cluster_access。这些属性在初始化的时候是由该源对应的站点设置的，也就是说，站点可以修改设置让别的源（不同源）能够通过JS脚本的方式访问自己的文档和属性。

```
class DomWindow : public ExcutionContext  {
...
Member<Frame> frame _ ;
Member<Object> security _ origin _ ;
Member<String> security _ token _ ;
}
```

SecurityOrigin和SecurityToken

- 而为了方便同源的文档之间互相访问，chromium还在DomWindow中设置了一个字符串名为SecurityToken。这个SecurityToken是由SecurityOrigin生成的，也就是说，如果两个文档，或者说两个frame的源相同，那么他们的SecurityToken就相同，这能减少两个同源文档之间访问时判断访问权限的分支，加快同源文档之间的访问速度。而如果两个SecurityToken不相同的文档在互相访问时，chromium就会进行更加完整的安全检测，这会在后面详细说明。

```
class SecurityOrigin  {
...
bool universal_access_;
bool cross_agent_cluster_access_;
}
```

WindowProxy和script_state

- 虽然chromium以及实现了DomWindow类来囊括一个文档的所有属性，但是使用这个类来进行安全检测过于冗杂，**为了提升性能**，chromium为frame实现了一个单独的窗口模型WindowProxy，专门用于处理context相关事宜。一个WindowProxy与一个frame唯一绑定，并提供了一个脚本运行所有需要的信息（包括v8:context，ExcutionContext，DOMWrapperWorld等等），script_state就是这些信息的封装。

```
class WindowProxy {
...
const Member<Frame> frame_;
Member<ScriptSatet> script_state_;
}
```

```
script_state_ = ScriptState:Create(context, world,
GetFrame()->DomWindow());
```

context的设置流程概览

- 初始化一个frame时，会先创建其对应的DomWindow，在这个过程中，SecurityOrigin和SecurityToken会先后设置；
- 然后将DomWindow视为ExcutionContext的具体实现，从快照中生成v8:context；
- 接着开始创建WindowProxy，在这个过程中，依然将DomWindow视为ExcutionContext，并将它和之前创建的v8:context同其他所需的信息一起封装进script_state中。

```
context =
V8ContextSnapshot::CreateContextFromSnapshot(isolate,
World(),&extension_configuration,global_proxy,document)
```

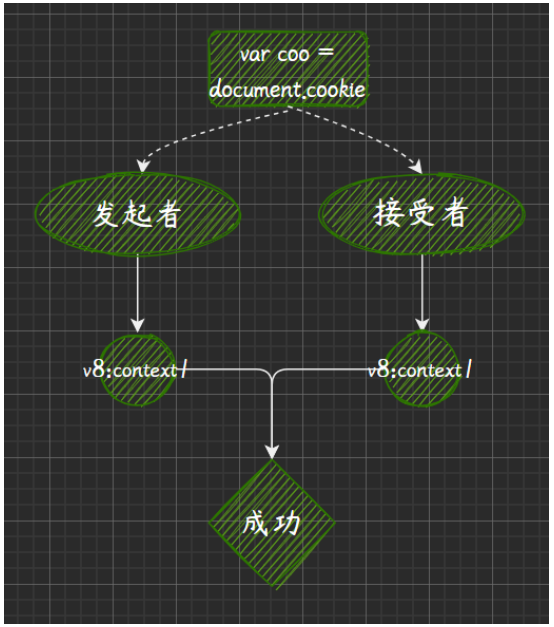
Extension_configuration 是使用DomWindow和其他一些变量生成的配置信息

访问权限的检测

- 不失一般性，我们只讨论“读”这个访问方式。我们将读取分成以下几种情况：
- 1.当前文档的JS代码访问当前文档的属性。
- 2.当前文档的JS代码访问另一个同源文档的属性。
- 3.当前文档的JS代码访问另一个不同源文档的属性。
- 基于同源策略的思想，在没有特殊设置的情况下，我们可以预见：前两种是可行的，第三种访问会被同源策略拒绝。在被访问站点有特殊设置的情况下，第三种访问是可以被允许的。

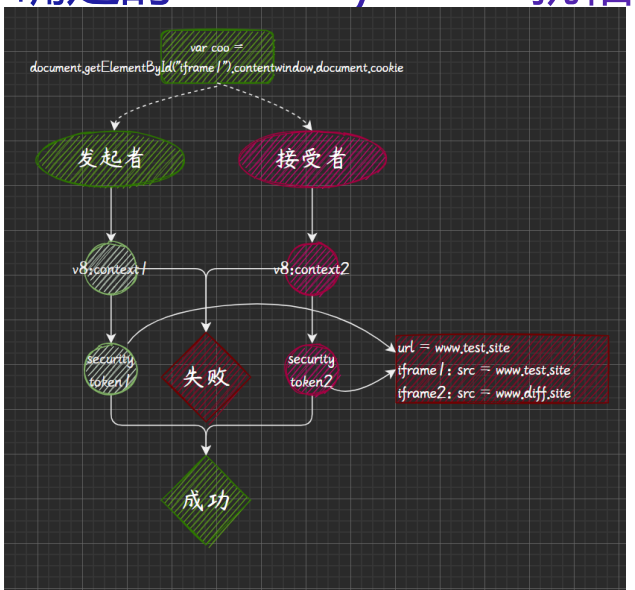
当前文档访问当前文档

- 在chromium种，当一个访问被发起时，会首先检测两个v8:context是否相同，因为v8:context由DomWindow唯一确定，而DomWindow由frame唯一确定，所以如果这个访问操作的发起者和接收者都是同一个frame，两者的v8:context必定相同，先检测这个会为这种情况省去很多分支，大大加快访问速度。



当前文档访问另一同源文档

- 我们之前提到，为了加速同源文档之间的交互，chromium设置了一个名为SecurityToken的属性，而两个文档不相同，则它们的context也不相同，不能通过第一步的检测，chromium紧接着在第二步就检测了两个文档的SecurityToken。通过SecurityToken的设置过程来看，两个文档的源相同，则其SecurityOrigin相同，那么由SecurityOrigin唯一确定的SecurityToken就相同。



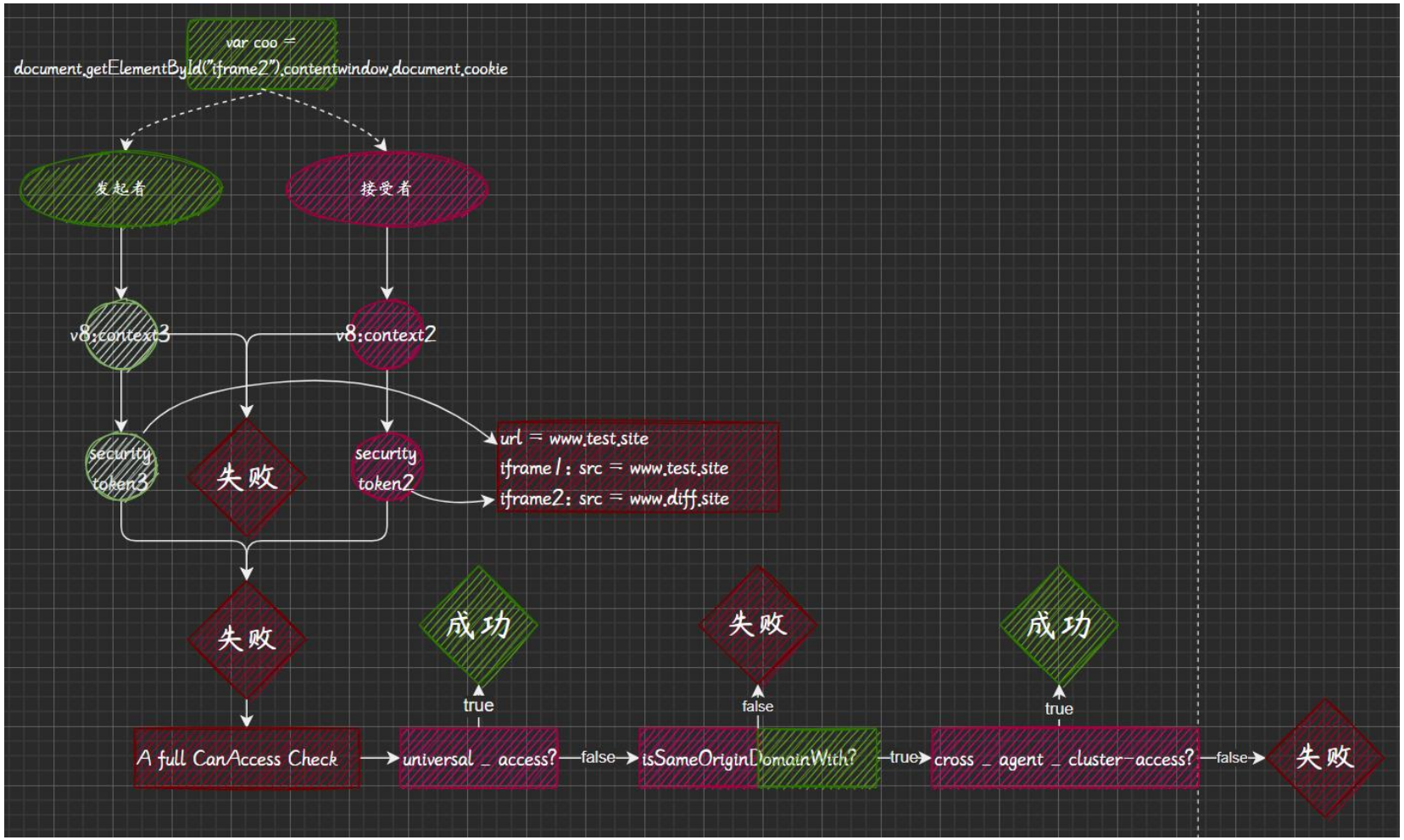
当前文档访问另一不同源文档

- 既然两个文档不同源，那么其SecurityToken就不可能相同，如果没有通过SecurityToken的判断，chromium就会进行一个完整的CanAccess检查，主要就是检查所谓的特殊设置。
- 一个站点如果设置了自身的属性可被其他站点的JS代码访问，那么之前提到的SecurityToken中的univsersal_access属性就会被置为真，如果这个站点设置了能被相同domain的源访问，比如https://source.chromium.org能访问https://bugs.chromium.org，那么cross_agent_cluster_access会被设置为真。

当前文档访问另一不同源文档

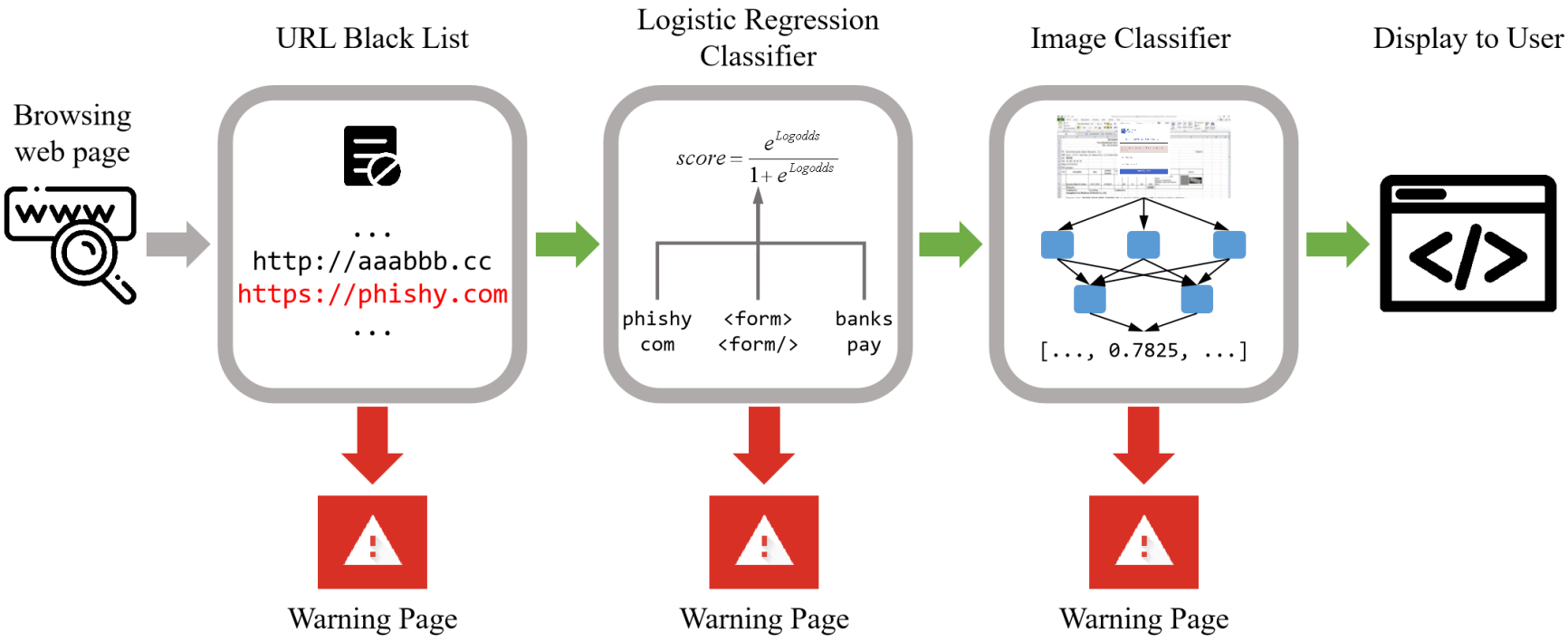
- 如果站点没有特殊设置，universal_access是默认值，为假，那么就会进行进一步的检测，首先chromium会查看访问者与被访问者的domain是否相同，如果相同，再判断该站点是否设置了能被相同domain的源访问（也就是检查cross_agent_cluster_access），在默认情况下，这个值为假，所以访问失败。
- 如果站点有特殊设置，比较弱的设置为只能被相同domain的源访问，那么第一个universal_access的判断就不会通过，会进入到后面的判断；比较强的设置为能被所有源访问，那么在第一个universal_access判断为真时，就能直接成功，不需要进入后面的判断。

当前文档访问另一不同源文档



12.2 钓鱼网页检测

- Google钓鱼网页检测机制包含3个部分：
  URL黑名单 + Logistic分类器 + 网页图像分类器



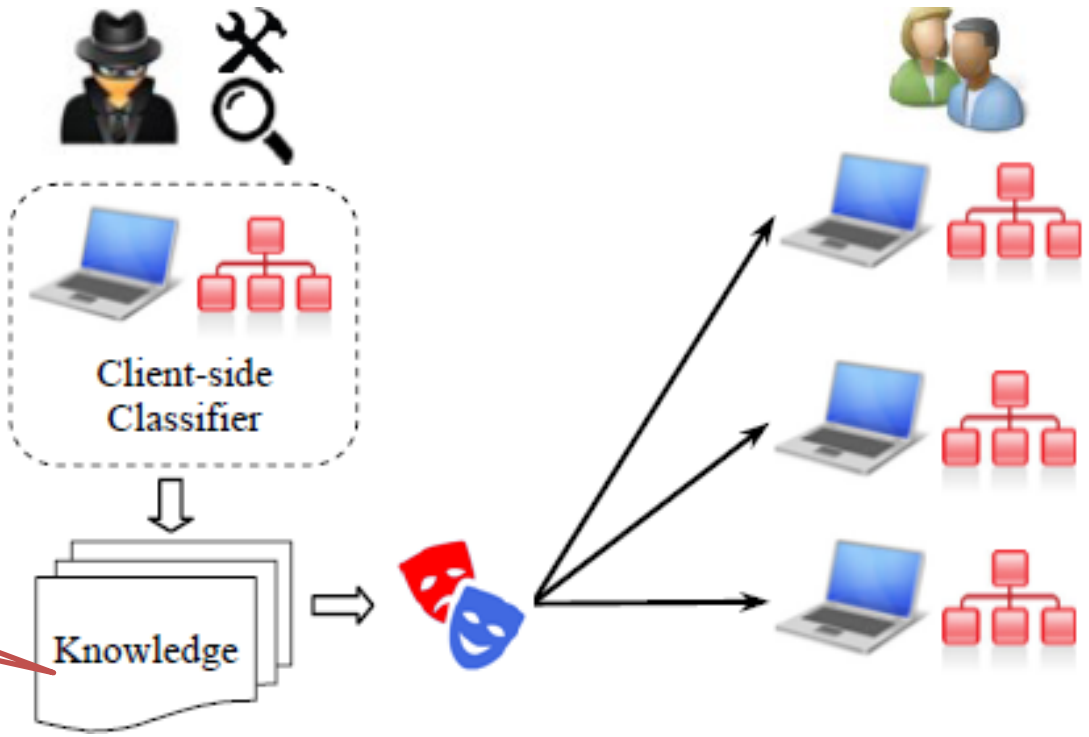
Logistic分类器

- 有的安全分类器被部署在客户端环境中(***client-side classifier***)，但其安全性并未得到应有的重视 (其具有更大的攻击面)
- 现实世界中得到广泛应用的商业分类器的安全性不明

Bin Liang, Miaoqiang Su, Wei You, Wenchang Shi, Gang Yang. *Cracking Classifiers for Evasion: A Case Study on the Google's Phishing Pages Filter*. In Proceedings of the 25th International World Wide Web Conference (**WWW 2016**). **CCF-A**

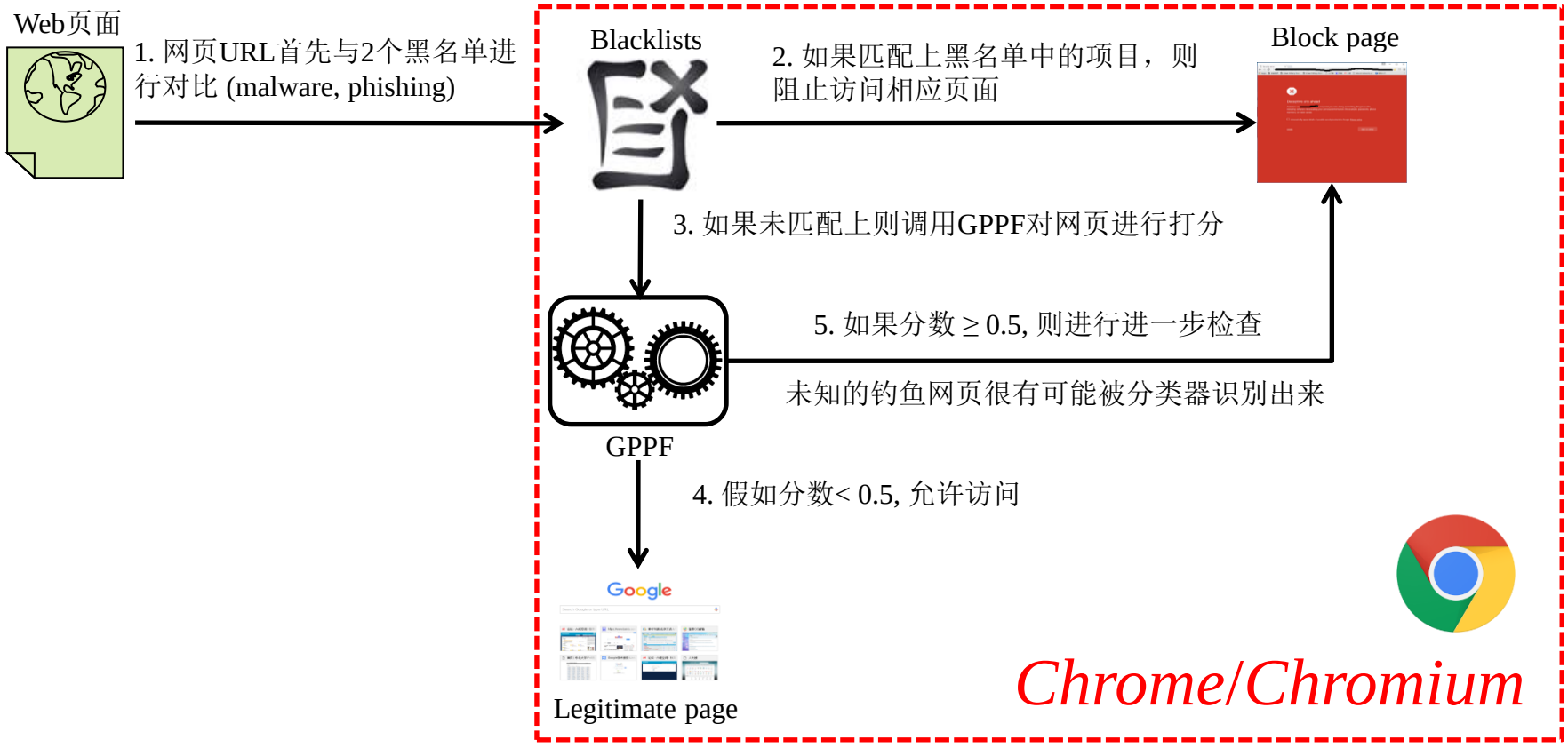
分类器破解

- 调试 Debugging
- 反汇编 Disassembling
- 代码分析 Code analysis
- 动态污点分析 Dynamic taint tracking
- …

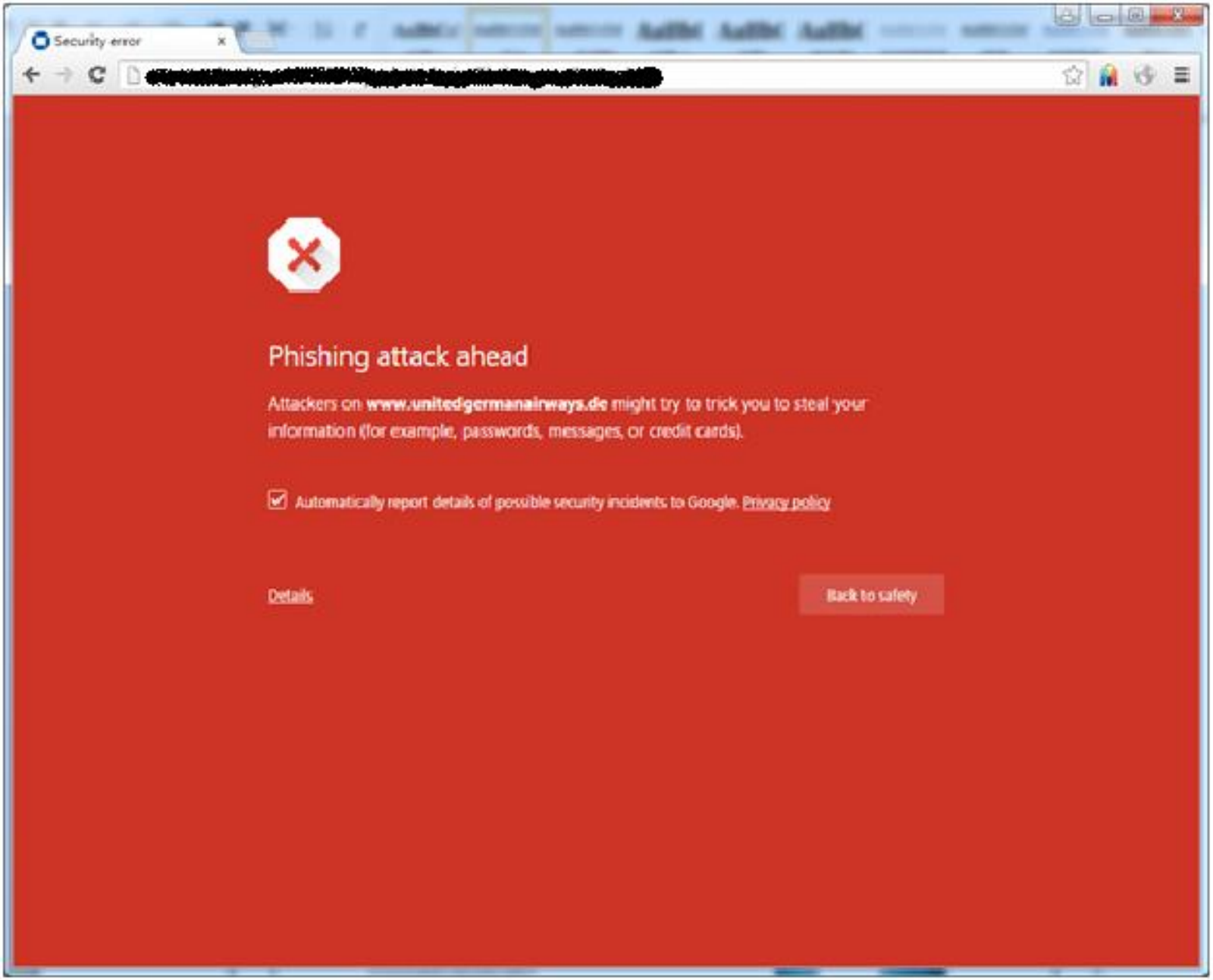


- 分类算法
- 分类特征
- 权重
- …

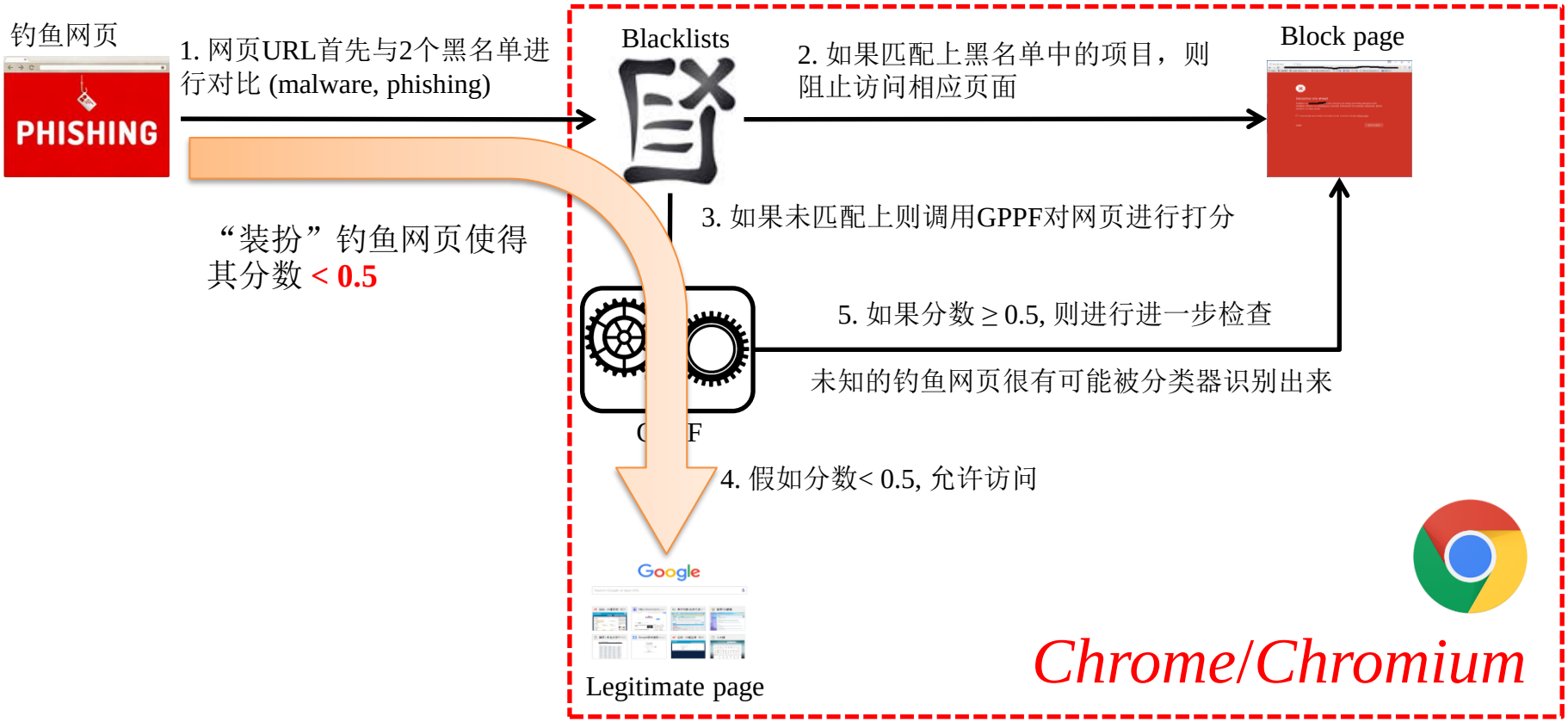
目标: Google钓鱼网页分类器(GPPF)



- Google 收集了海量的页面来训练GPPF
- **超过10亿用户** (被集成在Google的 *Chrome*/*Chromium*浏览器中)
- 每天检查数以十亿计的URL
- 每周能够检测出超过 60,000个新钓鱼网站


Security error
Phishing attack ahead
Attackers on www.unitedgermanairways.de might try to trick you to steal your information (for example, passwords, messages, or credit cards).
Automatically report details of possible security incidents to Google. Privacy policy
Details
Back to safety

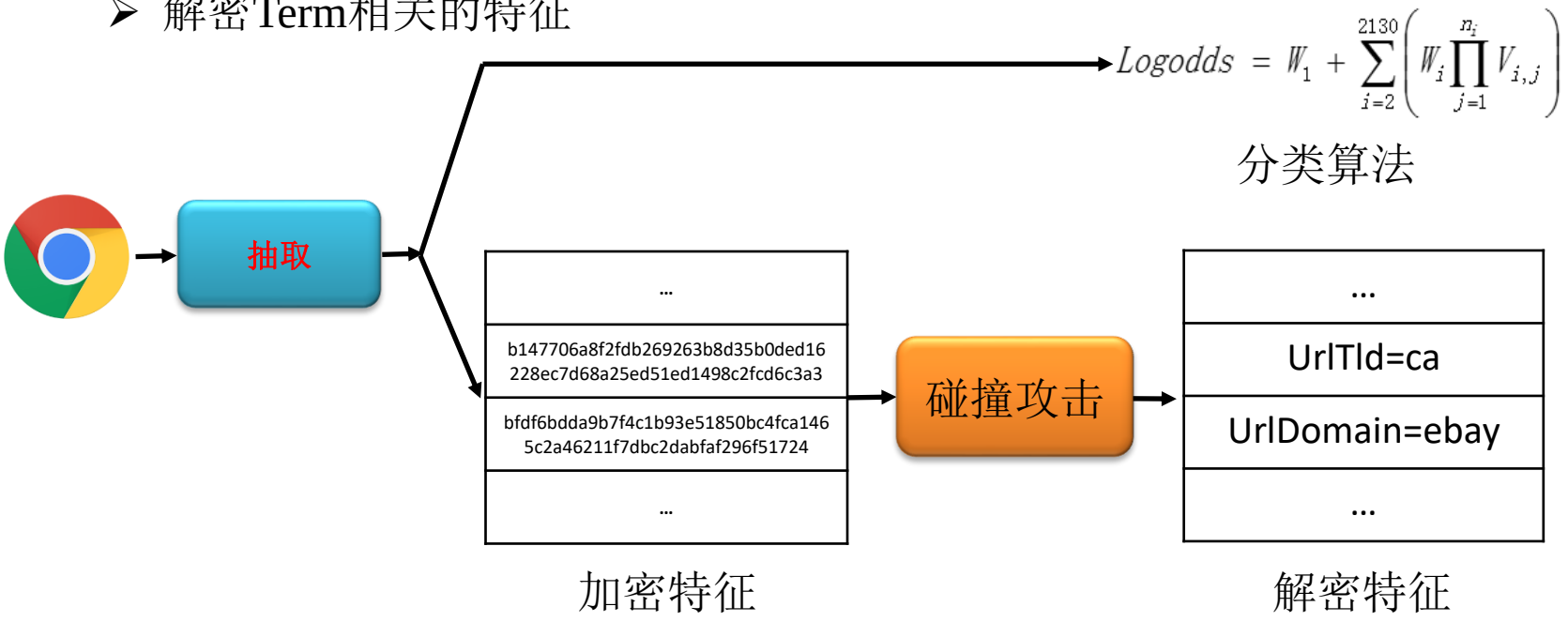
绕开 GPPF



驱使*Chrome/Chromium* 将一个**钓鱼页面**误分类为一个**合法页面**

破解 GPPF

- 抽取加密后的分类模型
  - 分类算法 Classification algorithm
  - 加密的模型特征 Encrypted model features
- 碰撞攻击
  - 解密URL相关的特征
  - 解密Term相关的特征



分类算法

通过采用一系列逆向工程技术，我们可知GPPF为一个Logistic回归分类器（**logistic regression classifier**）：

$$Logodds = W_1 + \sum_{i=2}^{2130}\left(W_i \prod_{j=1}^{n_i} V_{i,j}\right) \quad (1)$$

$$score = \frac{e^{Logodds}}{1 + e^{Logodds}} \quad (2)$$

- **2,130** 规则（R_1~R_{2130}）
  - 每个规则有一个正的或负的权重 (W_1~W_{2130})
  - 每个规则包含1到4个特征
- **1,009** 特征
- 假若 score **≥ 0.5**, 页面将被分类为一个潜在的钓鱼页面，否则为一合法页面

分类模型

- Google对GPPF特征使用 SHA256 哈希算法进行了加密

规则	特征数目	加密特征 (SHA-256)	权重
Rule1493	2	32ffbec120ed857f57f3d7bb37e6652955b21da7a7efd81d9a9aa2865173eb35	-1.26907706
		ec92914c7db4483437c849758c45cf8bbc6dd0148cdb2f72bec0a728e8c91a7d	
Rule2050	1	760e98536a709d0fcb9b717eb542cc5af77bbabf60a501dbf7df81a111d1e807	2.5238471

- 对于每一个规则通过GDB调试器抽取出其权重和所含特征数目
- 根据特征数目用脚本从内存中抽取出加密特征

URL 特征

No.	Page URL Features	Model URL Features
1	The hostname is an IP address?	*UrlHostIsIpAddress*
2	The number of other host components is greater than one?	*UrlNumOtherHostTokens>1*
3	The number of other host components is greater than three?	*UrlNumOtherHostTokens>3*
4	Top level domain	*UrlTld=**
5	The first host component below top level domain	*UrlDomain=**
6	Other host components	*UrlOtherHostToken=**
7	Path token in URL	*UrlPathToken=**

...
b147706a8f2fdb269263b8d35b0ded16228ec7d68a25ed51ed1498c2fcd6c3a3
bfdf6bdda9b7f4c1b93e51850bc4fca1465c2a46211f7dbc2dabfaf296f51724
...

DOM 特征

No.	Page DOM Features	Model DOM Features
1	Page has <form> element?	*PageHasForms*
2	Page has <input type=text> element?	*PageHasTextInputs*
3	Page has <input type=password> element?	*PageHasPswdInputs*
4	Page has <input type=radio> element?	*PageHasRadioInputs*
5	Page has <input type=checkbox> element?	*PageHasCheckInputs*
6	The number of <script> elements in the page is greater than 1?	*PageNumScriptTags>1*
7	The number of <script> elements in the page is greater than 6?	*PageNumScriptTags>6*
8	Token feature containing each external domain that is linked to	*PageLinkDomain=**
9	Fraction of form elements whose action attribution points to an external domain	*PageActionOtherDomainFreq*
10	Fraction of links in the page which point to an external domain	*PageExternalLinksFreq*
11	Fraction of page links that use https	*PageSecureLinksFreq*
12	Fraction of images whose src attribution points to an external domain	*PageImgOtherDomainFreq*

Term 特征

- **432** 个术语（Term）相关特征
- 特征形式：

  “**PageTerm= word$_1$ … word$_i$**”
- GPPF会从网页中的文字中抽取字词，并使用**murmurhash3算法**来构建候选词

…
6ebe7e49487ceb89dd5c12b47010336c dd4046dc1277eb5664ad56230ecc3ab2
ecbc6dd7bd276c61c529cb99d2ae40a4 2386edc8fd578dce1c790f340c2a9085
…

解密 URL相关特征

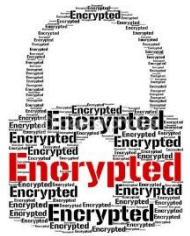
8,000 top level domain names

30,000+ history phishing URLs

Legitimate URLs in thousands of top sites

2,000,000+ domain names

“ UrlTld=***com*** ”



Encrypted URL features

607860731**50**C38692E9C67202D396386D
46CD5CD9D544B4055DC84B195247C39

607860731**50**C38692E9C67202D396386D
46CD5CD9D544B4055DC84B195247C39

碰撞攻击

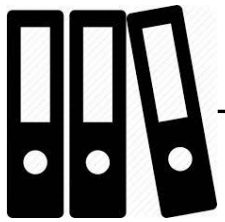
解密后的特征 “ UrlTld=***com***”

解密 URL相关特征

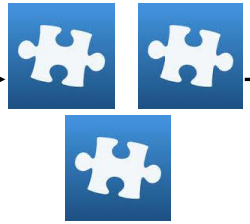
- **563** 个加密了的URL相关特征
- **426 (75.7%)** 个被成功解密
- 耗时: **0.5** 小时

加密特征(SHA256)	解密结果
…	…
b147706a8f2fdb269263b8d35b0ded16 228ec7d68a25ed51ed1498c2fcd6c3a3	UrlTld=ca
bfdf6bdda9b7f4c1b93e51850bc4fca1 465c2a46211f7dbc2dabfaf296f51724	UrlDomain=ebay
…	…

使用语料库解密Term相关特征



Several corpora in seven popular languages (**English, French, German, Spanish, Dutch, Chinese, Japanese**)



Candidate words

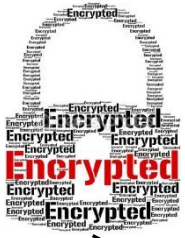
Candidate words:
“ username”
“password”

Terms

Terms:
“ PageTerm=username ”
“ PageTerm=password”
“ PageTerm=username password”

PageTerm=username
0870B1B45AD7364951F2D82CEA1F8DB2
05B657D61A577EF2C1F5554886F95663
PageTerm=password
26467A82A635C3800A02FAB20E795085
789EA6B6033C8CC180E00A480DB7E132
PageTerm=username password
FCFCA8E6DBDB3E49E1CEE8C6CBCB6A
B04849E6FDA596830A716572310C5B2304

Encrypted terms



26467A82A635C3800A02FAB20E795085
789EA6B6033C8CC180E00A480DB7E132

FCFCA8E6DBDB3E49E1CEE8C6CBCB6A
B04849E6FDA596830A716572310C5B2304

碰撞攻击

解密后特征

“PageTerm=***password***”
“PageTerm=***username password***”

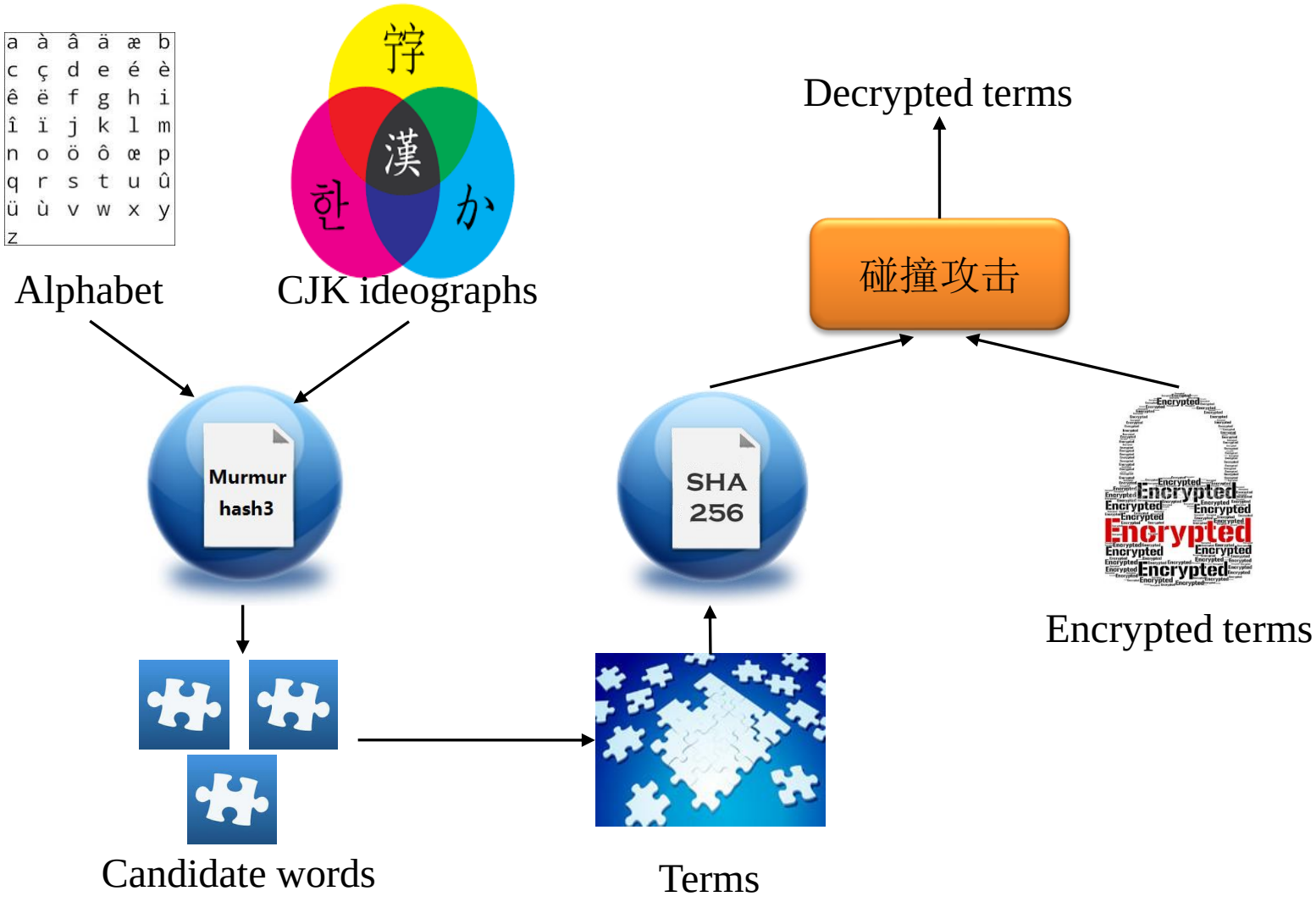
使用语料库解密Term相关特征

- **432** 个Term相关特征
- **292 (69.7%)** 在8.8小时内被成功解密

Language	Decrypted	Time
English	201	1.7 hours
French	6	2.3 hours
German	51	3.2 hours
Spanish	5	1.1 hours
Dutch	1	6 minutes
Chinese	27	20 minutes
Japanese	1	5 minutes
Sum	**292**	**8.8 hours**

加密特征(SHA256)	解密结果
…	…
26467a82a635c3800a02fab20e795085 789ea6b6033c8cc180e00a480db7e132	PageTerm=password
Fcfca8e6dbdb3e49e1cee8c6cbcb6ab0 4849e6fda596830a716572310c5b2304	PageTerm=username password
…	…

暴力破解解密Term相关特征



暴力破解解密Term相关特征

- **281 (65.0%)** 被成功解密
- 耗时: 大概 **16** 小时

Term Size	Candidate Words	Decrypted	Time
1-word	1-letter to 8-letter	186	1 minute
2-word	1-letter to 8-letter	76	8.6 hours
3-word	1-letter to 6-letter	15	14 minutes
4-word	1-letter to 4-letter	4	7.4 hours
sum		**281**	**16.25 hours**

Term Size	Candidate Words	Decrypted	Time
1-word	1-ideograph to 3-ideograph	31	1 minute
2-word	1-ideograph to 3-ideograph	7	< 1 minute
3-word	1-ideograph to 3-ideograph	2	< 1 minute
4-word	1-ideograph to 3-ideograph	0	2 minute
Sum		**40**	**5 minutes**

加密特征(SHA256)	解密结果
…	…
041c20385f825daacacc60d71604ff23beb656dc990b8c9a523fc3cd4705eb78	PageTerm=支付 密码
6526786532d03f01b2d48c32ed16a18f1d8c221abf7bf863cffd80c5d797060a	PageTerm=débito
b2fd98c71610ac72724aadc7f5d4ae7052240fa1350eafef0323d48542660e56	PageTerm=カード
…	…

解密结果

- 破解结果
  - **1009** 个模型特征（**801**个被解密）

Category	Model Features	Total	Decrypted		%
URL-related	UrlTld=*	563	69	426	75.7%
	UrlDomain=*		21		
	UrlOtherHostToken=*		28		
	UrlPathToken=*		201		
	PageLinkeDomain=*		107		
Term-related	PageTerm=*	432	375		86.8%
Sum		**995**	**801**		**80.5%**

  - **2130** 个规则
    - **1807(84.8%)** 个规则被完全解密
    - **196 (9.2%)**个规则被部分解密
    - **127 (6.0%)**个规则未被解密

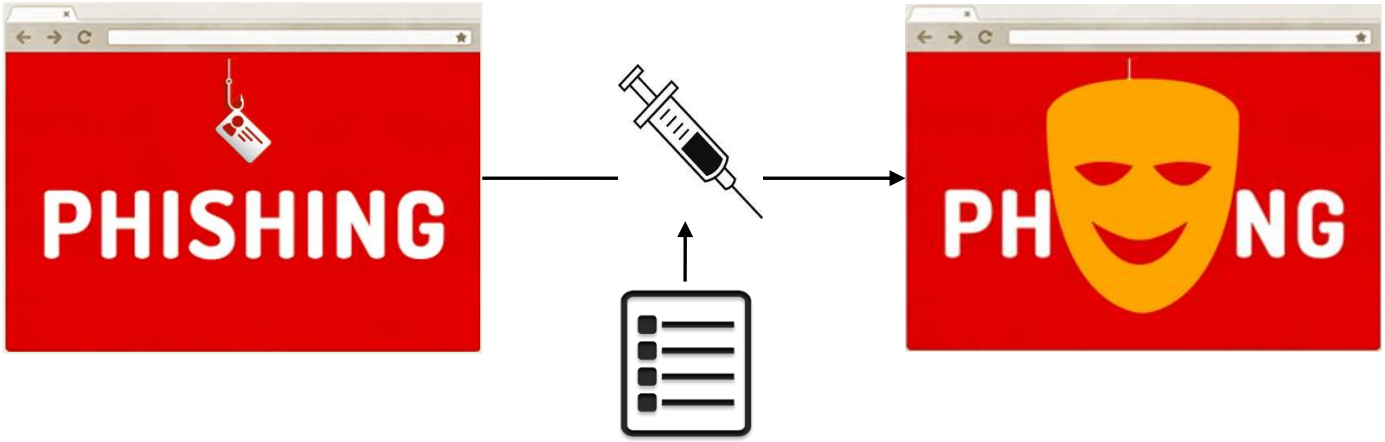
足够发起绕开攻击！

绕开攻击

- 插入“好”特征（**Good Features Insertion**）：在目标网页中插入能够对最终评分提供“负贡献”的特征（对应规则权重为负）

- 删除“坏”特征（**Bad Features Elimination**）:在目标网页中删除能够对最终评分提供提供“正贡献”的特征（对应规则权重为正）

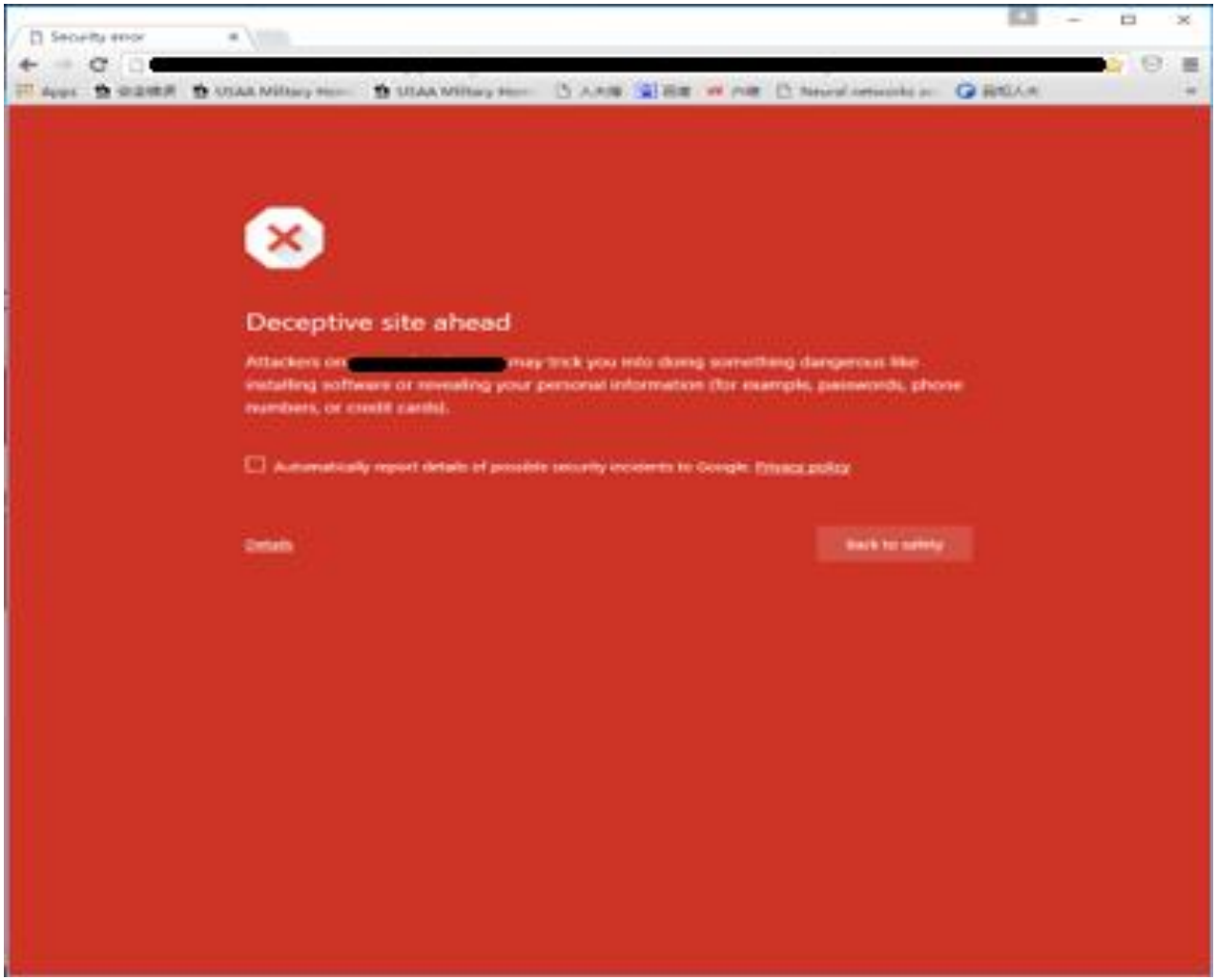
插入“好”特征



- 选择只出现在排序靠前的负规则中的特征
- 成功地扰动了所有选自PhishTank 的100个钓鱼网页(**100%**)

Feature	MIN	MAX	Average
URL	1	10	2.5
DOM	1	6	2.2
Term	1	17	3.7

演示



评分: **0.985543**

演示


USAA Military Home, Life & Auto Insurance _ Banking & Investing_term.html
Auto Insurance
</h2>

Count on USAA for affordable auto insurance. Save an average of $450 a year.

<a class="primary" href="https://www.usaa.com/inet/gas_pc_pas/GyMemberAutoHistoryServlet?action=INIT&link_type=QuoteHistory&link_type=PurchaseQuote&ac

<a class="secondary" href="https://www.usaa.com/inet/pages/auto_insurance_main?offerName=pubHomePro_PrdBckt_1_030512_PnC_AutoIns_learnmore">Learn More<span class="hid

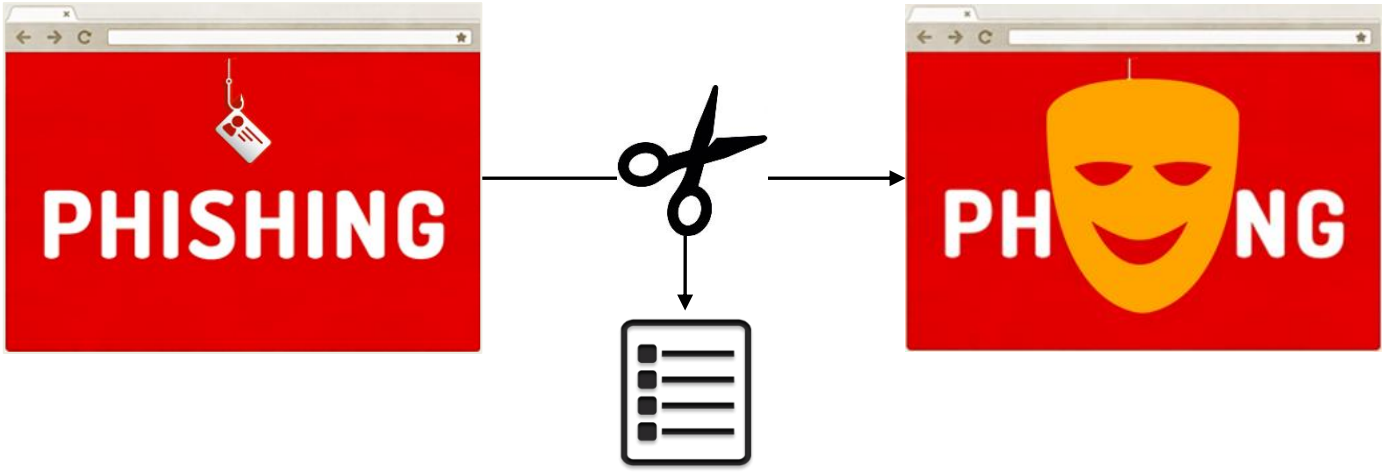
<a class="secondary" href="https://www.usaa.com/inet/ent_membereligibility/CpMemberEligibility?action=executetask&target=BuyQuote&offerName=pubHomePro_PrdBckt

插入15个“好”词

演示

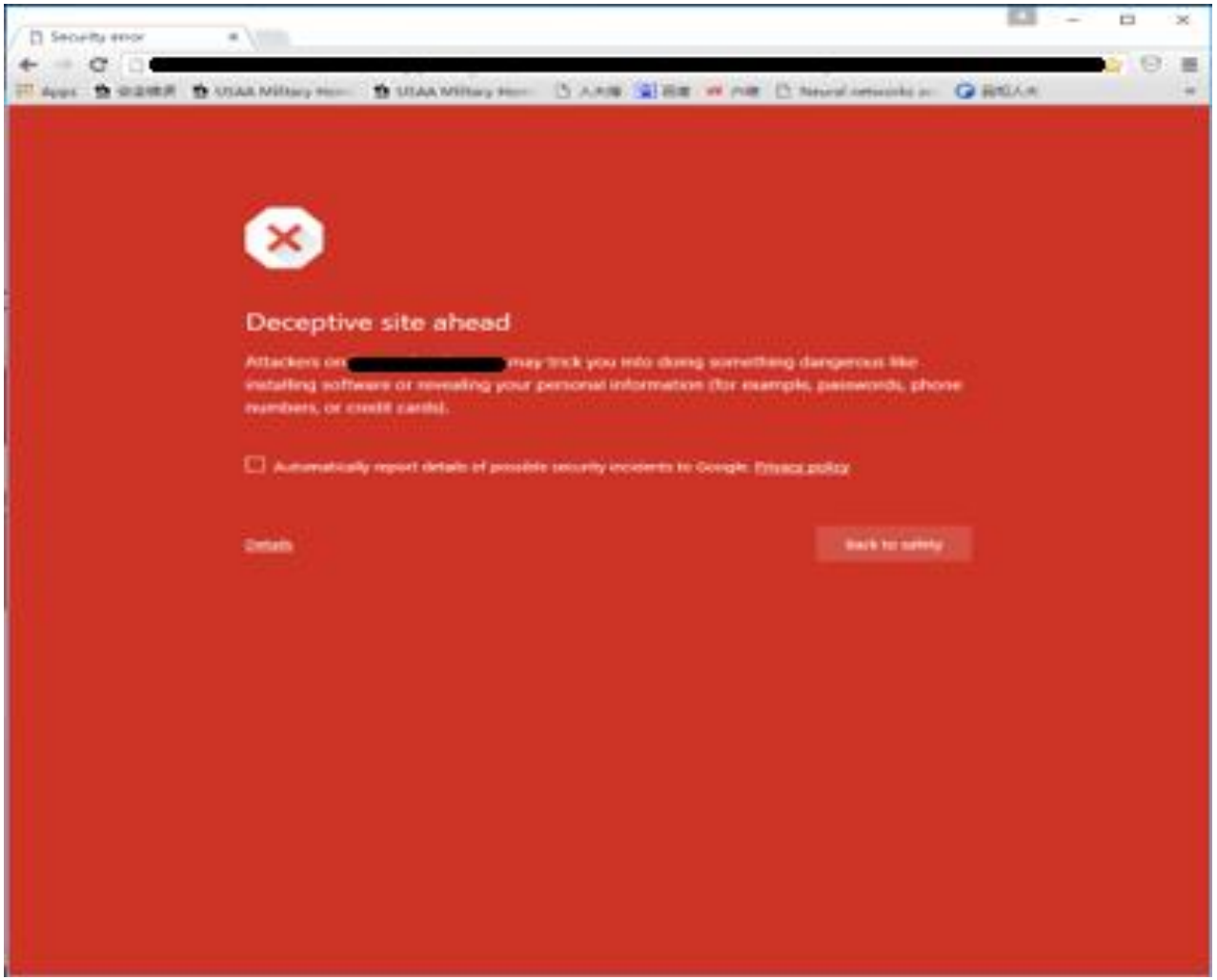
评分： **0.264653**

删除“坏”特征



- 使用一个基于搜索的方法在目标网页里发现待删除的内容
- 成功地扰动了所有选自PhishTank 的100个钓鱼网页(**100%**)，最多只需删除5个坏特征

演示



评分: **0.985543**

演示

```
<h1 class="headerUpdateH1 prepend-2 prepend-top-pad-6">For auto insurance, free checking, credit cards  investments and more, let us serve you.</h1>

<h1 class="headerUpdateH1 prepend-2 prepend-top-pad-6">For auto insurance, free checking, credit card  investments and more, let us serve you.</h1>
```

PageTerm=cards → PageTerm=card

```
<div class="login_container_left">
    <label class="label" for="usaaNum"><span class="hiddenMessage">USAA </span>Online ID</label>
    <input name="userid" id="usaaNum" class="login_input pie input_rounded_corners" size="25" maxlength="20" onkeydown="jChangeFocus(event);" onkeypress="removeErrorMe
</div>

<div class="login_container_left">
    <label class="label" for="usaaNum"><span class="hiddenMessage">USAA </span>Online lD</label>
    <input name="userid" id="usaaNum" class="login_input pie input_rounded_corners" size="25" maxlength="20" onkeydown="jChangeFocus(event);" onkeypress="removeErrorMe
</div>
```

PageTerm=ID → PageTerm=lD

```
<span class="forgotten_container yuimenubaritemlabel">
<div>
    <div class="fl"><a class="authentication_link asText" href="https://www.usaa.com/inet/ent_proof/proofingEvent?action=Init&amp;event=forgotOnlineId&amp;wa_ref=pub_auth_nav_

    <div class="fl">or <a class="authentication_link" href="https://www.usaa.com/inet/ent_proof/proofingEvent?action=Init&amp;event=forgotPassword&amp;wa_ref=pub_auth_nav

    <div class="fl"><span class="left-aligned background divider"><a class="authentication_link" href="https://www.usaa.com/inet/ent_proof/proofingEvent?action=Init&amp;event=
</div>
</span>

<span class="forgotten_container yuimenubaritemlabel">
<div>
    <div class="fl"><a class="authentication_link asText" href="http://www.usaa.com/inet/ent_proof/proofingEvent?action=Init&amp;event=forgotOnlineId&amp;wa_ref=pub_auth_nav_f

    <div class="fl">or <a class="authentication_link" href="http://www.usaa.com/inet/ent_proof/proofingEvent?action=Init&amp;event=forgotPassword&amp;wa_ref=pub_auth_nav_

    <div class="fl"><span class="left-aligned background divider"><a class="authentication_link" href="http://www.usaa.com/inet/ent_proof/proofingEvent?action=Init&amp;event=
</div>
</span>
```

PageSecureLinksFreq

删除3个“坏”特征

演示

评分: **0.056245**

未来: 深度学习（Deep Learning）

Google已经开始使用深度神经网络（deep neural network）来检测SPAM页面

Google Turns to Deep Learning Classification to Fight Web Spam

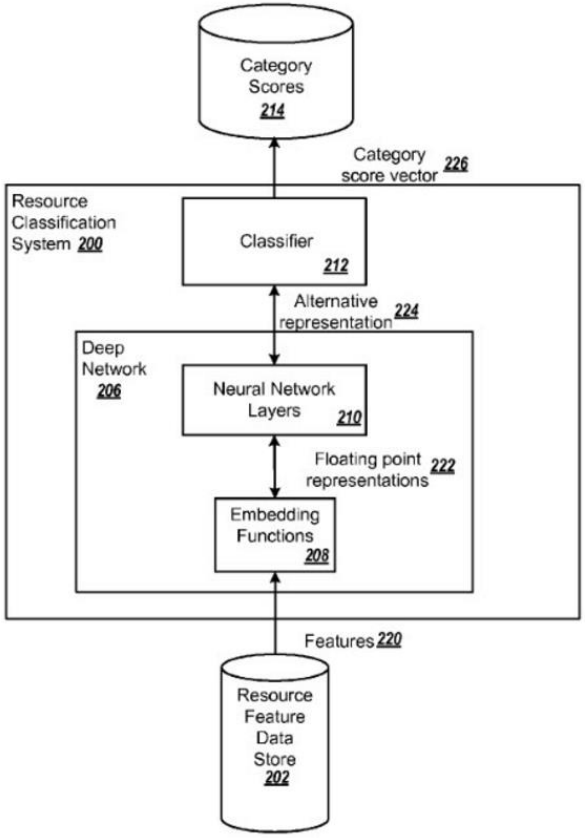
09/24/2014 by Bill Slawski

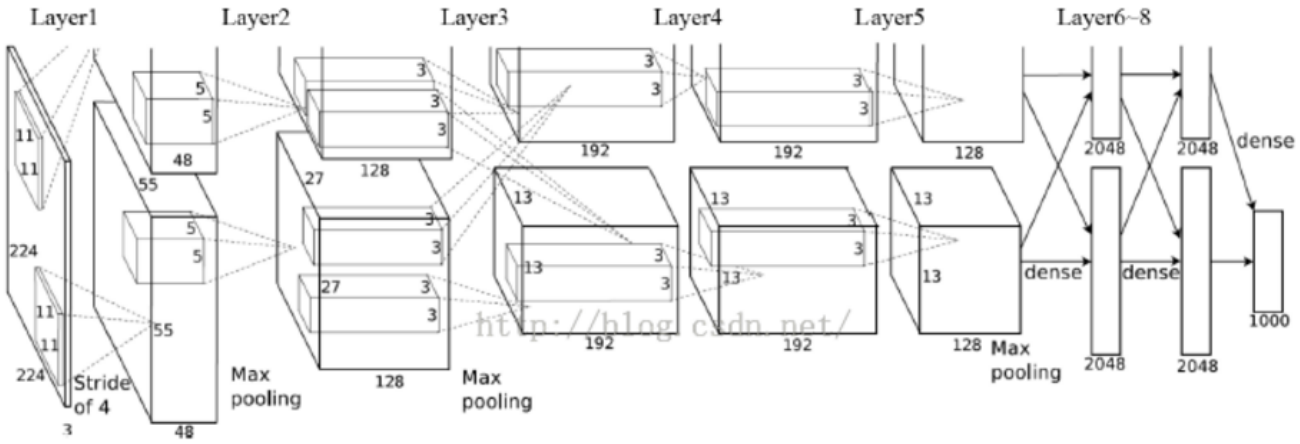
Facebook 33 Twitter 181 Google+ 92
LinkedIn 57 Pinterest 0

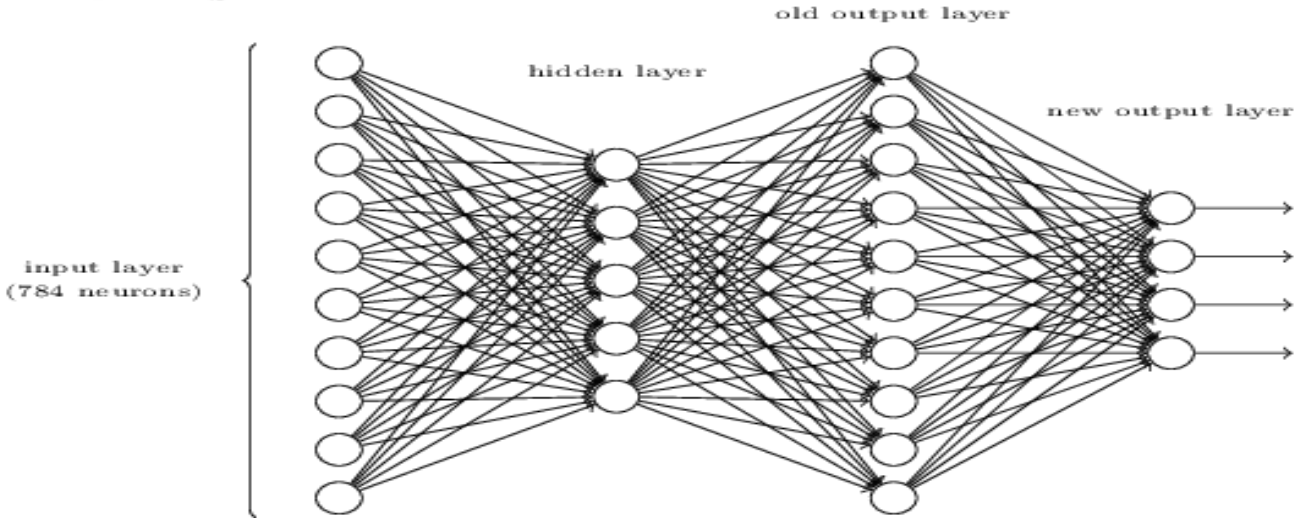
363 shares

In the past few years, Google has been busy building what has become known as the Google Brain team, which started out by having its deep learning approach watching videos until it learned to recognize cats.

Google has been hiring a number of people to add to the abilities of their deep learning team, including a pricy acqui-hire in the UK earlier this year, as described in More on DeepMind: AI Startup to Work Directly With Google's Search Team



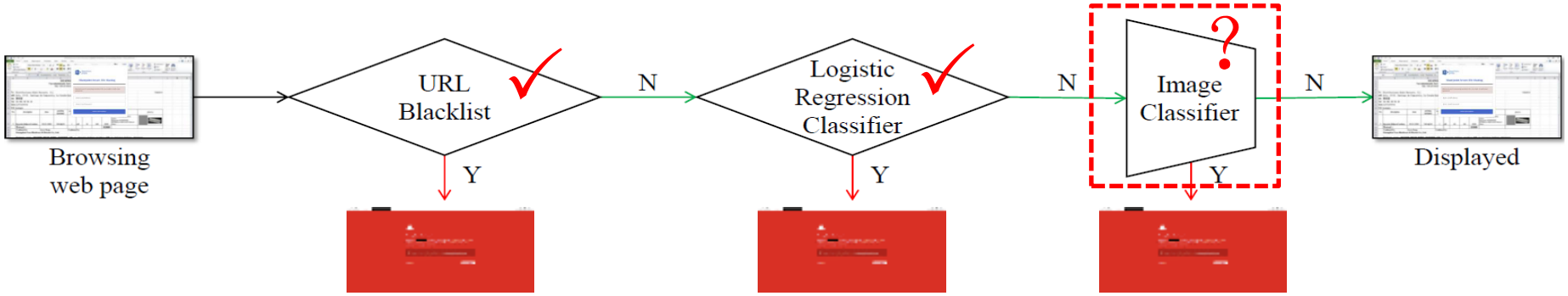

Layer1
Layer2
Layer3
Layer4
Layer5
Layer6~8
224
11
11
Stride
of 4
48
55
5
5
Max
pooling
27
128
3
Max
pooling
192
13
128
Max
pooling
2048
dense
1000


old output layer
hidden layer
new output layer
input layer
(784 neurons)

复杂、非线性

新一代Google钓鱼网页分类器攻击

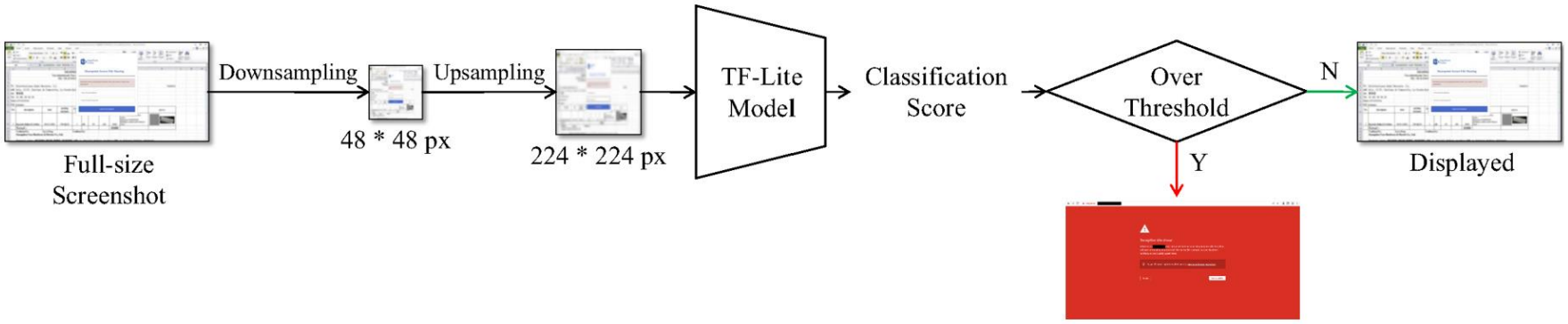
- 最近Google升级了Chrome中的钓鱼网页分类器，引入了一个基于深度学习的图像分类器来检测钓鱼网页。还能绕开吗？**能！**



Changqing Miao, Jianan Feng, Wei You, Wenchang Shi, Jianjun Huang, and Bin Liang. ***A good fishman knows all the angles: A critical evaluation of google's phishing page classifier***. In Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (**CCS 2023**)

Google新一代钓鱼网页分类器工作流程

- 浏览器获得网页截图;
- 进行一次降采样 (48*48 px);
- 进行一次升采样 (224*224 px)
- 送入一个卷积神经网络模型（TF-Lite）；
- 得到19个类别的分类得分（18个钓鱼网页类别+1个正常网页类别）；
- 若某个钓鱼网页类别的得分超过设定阈值，则报告为钓鱼网页。

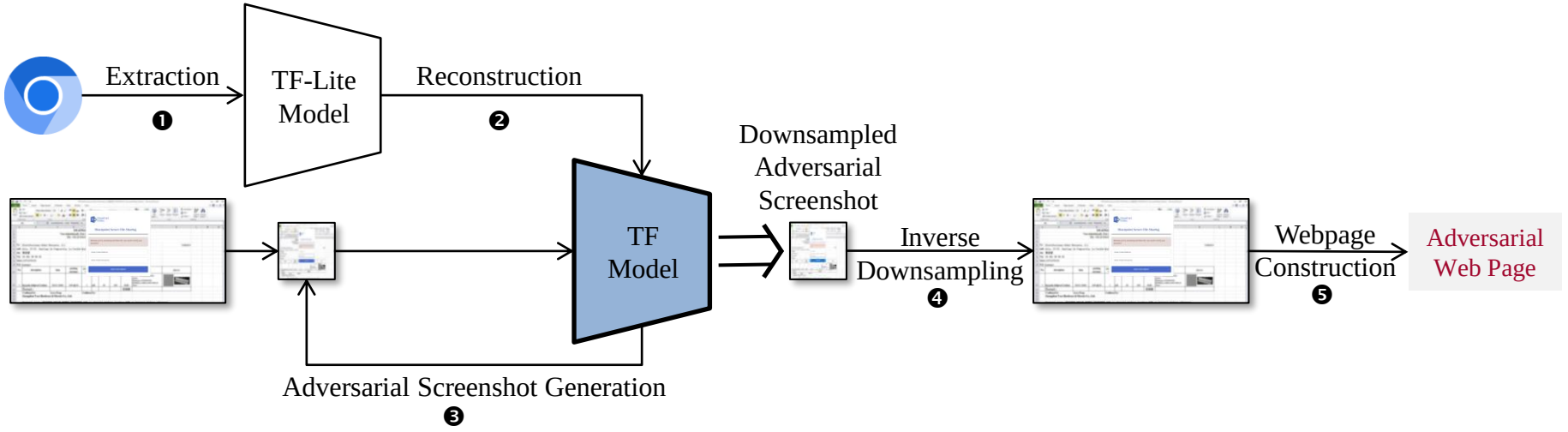


技术挑战

- 需要保证用户不易察觉，攻击**扰动尽量小**；
- 原始模型（**TF-Lite**）不适宜进行迭代优化；
- 分类模型的输入是一个降采样后的网页截图，由模型获得的对抗扰动需要通过**逆降采样**体现在网页里，同时保证不损失攻击效果；
- 生成全尺寸对抗样本的搜索空间巨大，需要保证**计算可行**。

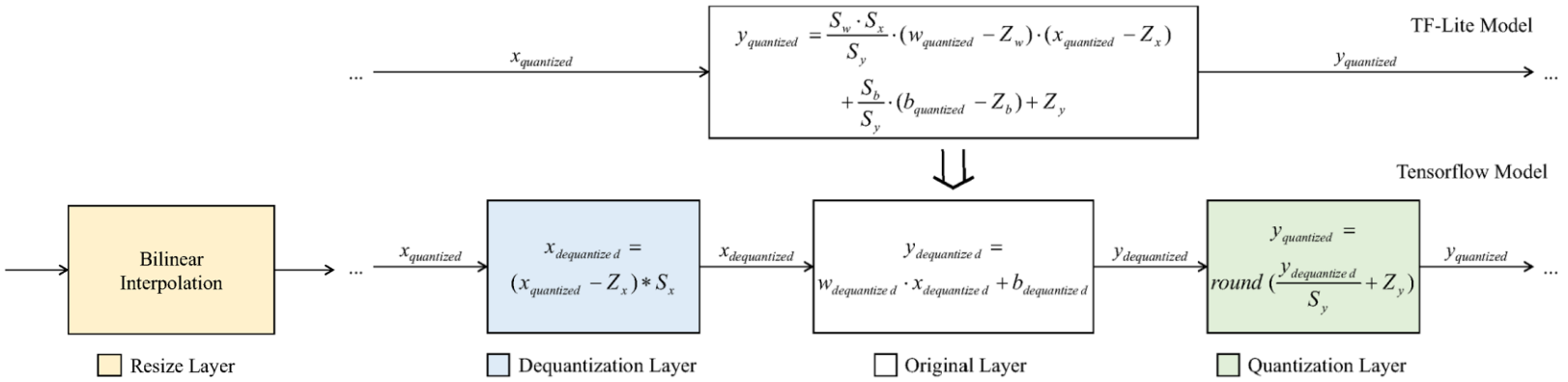
绕开方法

- **模型抽取：**从浏览器实现中获得模型；
- **模型重构：**将其转换为适宜优化的形式；
- **生成扰动：**通过迭代优化生成初始对抗攻击样本（小尺寸图像）；
- **逆降采样：**获得全尺寸攻击图像；
- **构建网页：**将全尺寸的对抗扰动植入目标网页中，获得对抗网页，绕开分类器检测。

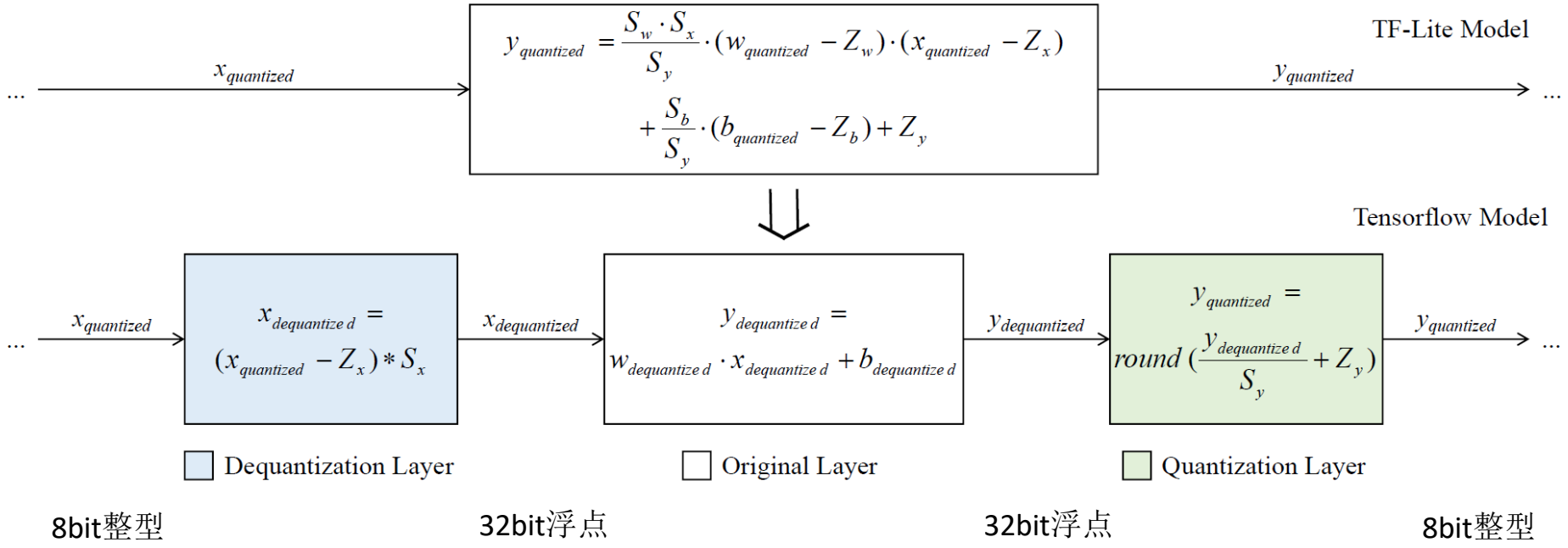


模型抽取与重构

- **模型抽取：**分析Chromium开源代码，使用调试器分析定位分类器相关代码，直接从运行时内存中提取模型数据，获得一个**TF-Lite模型**；
- **模型重构：**TF-Lite是一种面向终端部署的轻量级模型，其省略掉了执行模型运算之外的大部分非必要功能，难以支持迭代攻击必需的计算梯度功能。我们将其重构为一个标准的**TensorFlow**模型（Pilot实验表明2个模型精确度一致）：
  - 对于TF-Lite中的参数量化（用整数运算替代浮点数运算），将其转换为标准TensorFlow网络运算层，并在前后添加**Dequantization**和**Quantization**层，来模拟TF-Lite中的运算，以在标准TensorFlow环境中进行梯度计算；
  - 加入一个双线性插值层实现升采样。



模型抽取与重构


TF-Lite Model
$y_{quantized} = \frac{S_w \cdot S_x}{S_y} \cdot (w_{quantized} - Z_w) \cdot (x_{quantized} - Z_x) + \frac{S_b}{S_y} \cdot (b_{quantized} - Z_b) + Z_y$
$x_{quantized}$
$y_{quantized}$
Tensorflow Model
$x_{dequantized} = (x_{quantized} - Z_x) * S_x$
$y_{dequantized} = w_{dequantized} \cdot x_{dequantized} + b_{dequantized}$
$y_{quantized} = round(\frac{y_{dequantized}}{S_y} + Z_y)$
Dequantization Layer
Original Layer
Quantization Layer
8bit整型
32bit浮点
32bit浮点
8bit整型

生成初始对抗样本

- **输入：**降采样后的网页截图（48*48 px）；
- **输出：**能够绕过钓鱼网页图像分类器的降采样后的网页截图（48*48 px）。
- **方法：**
  - 综合**视觉显著性**（用一个预训练模型计算）和**分类贡献**评估选择候选像素进行扰动（优先选择视觉显著性低但分类贡献高的像素点），以尽量在不引起浏览器用户注意的情况下攻击成功;
  - 使用基于梯度计算的迭代优化，扰动修改候选像素，获得初始对抗样本。

像素全集

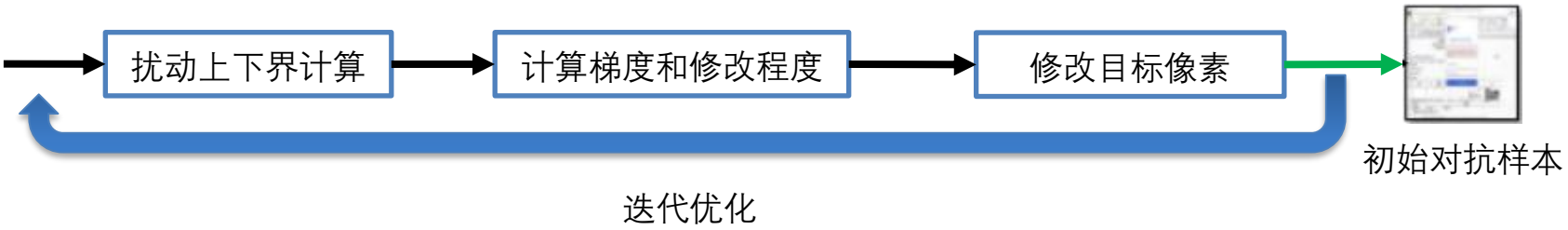
- **低视觉显著性**: 用户不容易察觉.
- **高分类贡献**: 扰动像素少且容易成功

$$Score\ (x) = contribution\ (x) - saliency\ (x)$$

Top k 候选像素

生成初始对抗样本

- **输入：**降采样后的网页截图（48*48 px）；
- **输出：**能够绕过钓鱼网页图像分类器的降采样后的网页截图（48*48 px）。
- **方法：**
  - 综合视觉显著性（用一个预训练模型计算）和分类贡献评估选择候选像素进行扰动（优先选择视觉显著性低但分类贡献高的像素点），以尽量在不引起浏览器用户注意的情况下攻击成功；
  - 使用基于梯度计算的迭代优化，扰动修改候选像素，获得初始对抗样本。



生成初始对抗样本

- 构建损失函数，计算候选像素相对于损失函数的梯度：

$$loss = cls_{benign} - \max(cls)$$

- 根据像素的显著性得分对像素分级，确定上下界**（显著性得分越高，扰动越受限）**；每次迭代修改程度与梯度大小相关**（梯度绝对值越大，扰动越大）**；

$$upper = \min(i+1, 255),\ lower = \max(i-2, 0);$$
$$\textbf{if } i \in H;$$
$$upper = \min(i+2, 255),\ lower = \max(i-4, 0);$$
$$\textbf{if } i \notin H \textbf{ and } saliency[i] > 0.5;$$
$$upper = \min(i+3, 255),\ lower = \max(i-6, 0);$$
$$\textbf{if } i \notin H \textbf{ and } saliency[i] \le 0.5;$$

$$D(g_x) = \frac{sign(g_x) * k_g * |g_x|}{\max(\max(g_{sign}), 0) + \varepsilon}$$

$$g_{sign} = \{|g_k| \mid sign(g_k) = sign(g_x)\}$$

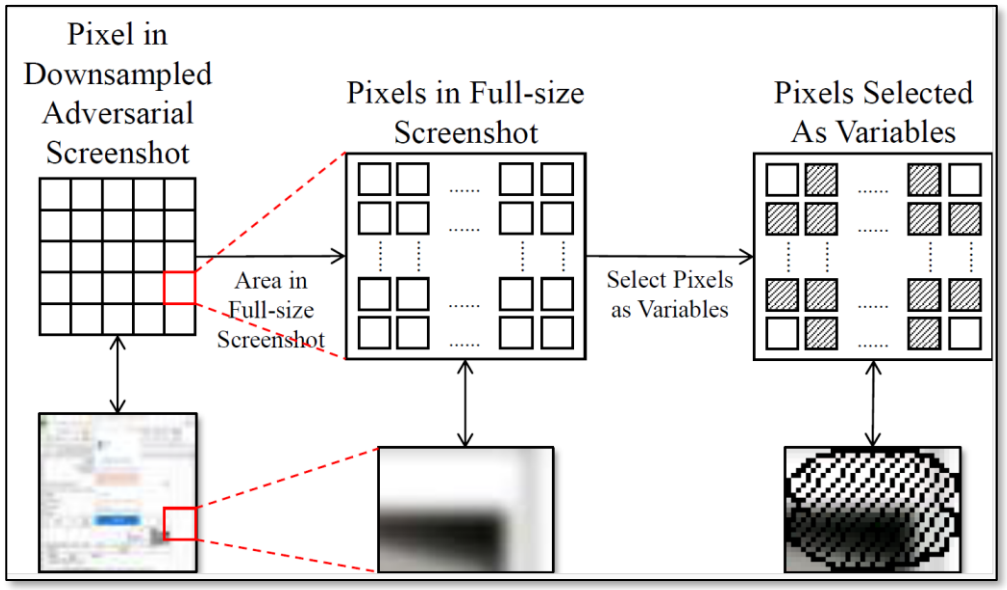
- 迭代过程：
  - 若攻击成功，则输出结果；
  - 若超越最大轮数，则攻击失败；
  - 若当前像素无法提升攻击效果，则新增候选像素；
  - 否则，则继续针对已有候选像素迭代。

逆降采样

- **输入：**初始对抗样本（48*48 px）
- **输出：**全尺寸对抗样本（1920*1080 px）
- **挑战：**生成全尺寸对抗样本，搜索空间巨大，组合爆炸会导致不可接受的计算量。
- **方案：**为缓解这一问题，将逆降采样问题转换为一个**整数线性规划问题**求解（使用GUROBI 优化器求解），涉及3个要素：
  - **决策变量：**逆降采样后体现在全尺寸攻击网页截图里的像素值；
  - **目标函数：**优化视觉可用性，减小逆降采样后像素值和原始网页截图中像素值的距离（**变动尽可能小**），并使得逆降采样后像素与周围像素相似（**变动看起来不太突兀**）；
  - **约束条件：**❶合法性约束：像素值范围合法（0~255的整数），保证正向降采样结果正确；❷视觉可用性约束：“摊平”像素值改变的影响，确保视觉可用性。

逆降采样——变量选择

- **选择策略：**使得扰动看起来像屏幕上的污渍。在扰动像素点所对应的全尺寸截图中像素点矩形区域中，选择内切椭圆区域内的像素点作为目标变量。



- **矩形区域**：边缘太尖锐，容易让浏览器用户察觉到异常；
- **椭圆区域**：边缘较平滑，晕染状的扰动可能让用户产生错觉，以为是屏幕上的污渍斑点。

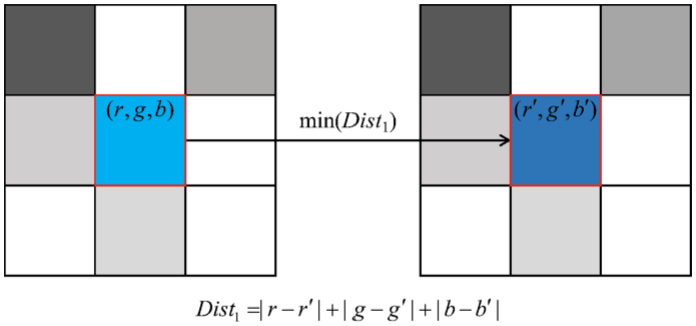
逆降采样——目标函数

$$
\begin{aligned}
\min \quad & Z = C_1^{\mathrm{T}} \cdot (U+V) + C_2^{\mathrm{T}} \cdot (M+N) \\
s.t. \quad & 0 \le (U-V) + P_{fullsized} \le 255 \\
& 0 \le U, V, M, N \le 255, \quad U, V, M, N : Integer \\
& (U-V) + P_{origin} = (M-N) + P_{blurred} \\
& P_{downsized} = K^{\mathrm{T}} \cdot (U - V + P_{fullsized}) \\
& \delta \le \textstyle\sum_c (U_c + V_c) \quad c = r, g, b
\end{aligned}
$$

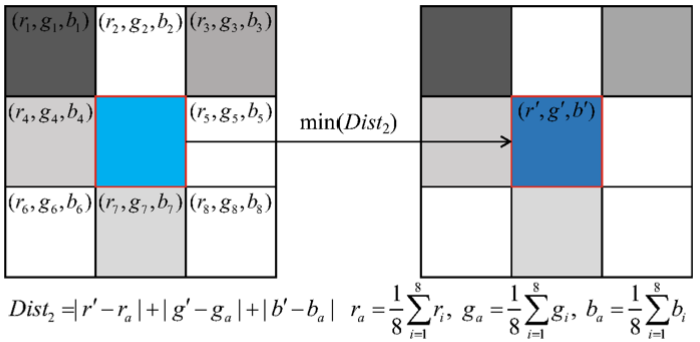
$$
\begin{aligned}
U &= \frac{|X - P_{origin}| + (X - P_{origin})}{2} \\
V &= \frac{|X - P_{origin}| - (X - P_{origin})}{2} \\
M &= \frac{|X - P_{blurred}| + (X - P_{blurred})}{2} \\
N &= \frac{|X - P_{blurred}| - (X - P_{blurred})}{2}
\end{aligned}
$$

逆降采样——目标函数

- 全尺寸对抗样本（截图）中扰动点与原始网页截图对应像素点的值间距离尽可能小（左图）；
- 扰动点与原始网页截图对应像素点周围的像素点之间尽可能相似（右图）。



$|x_{adv} - x_{origin}|$

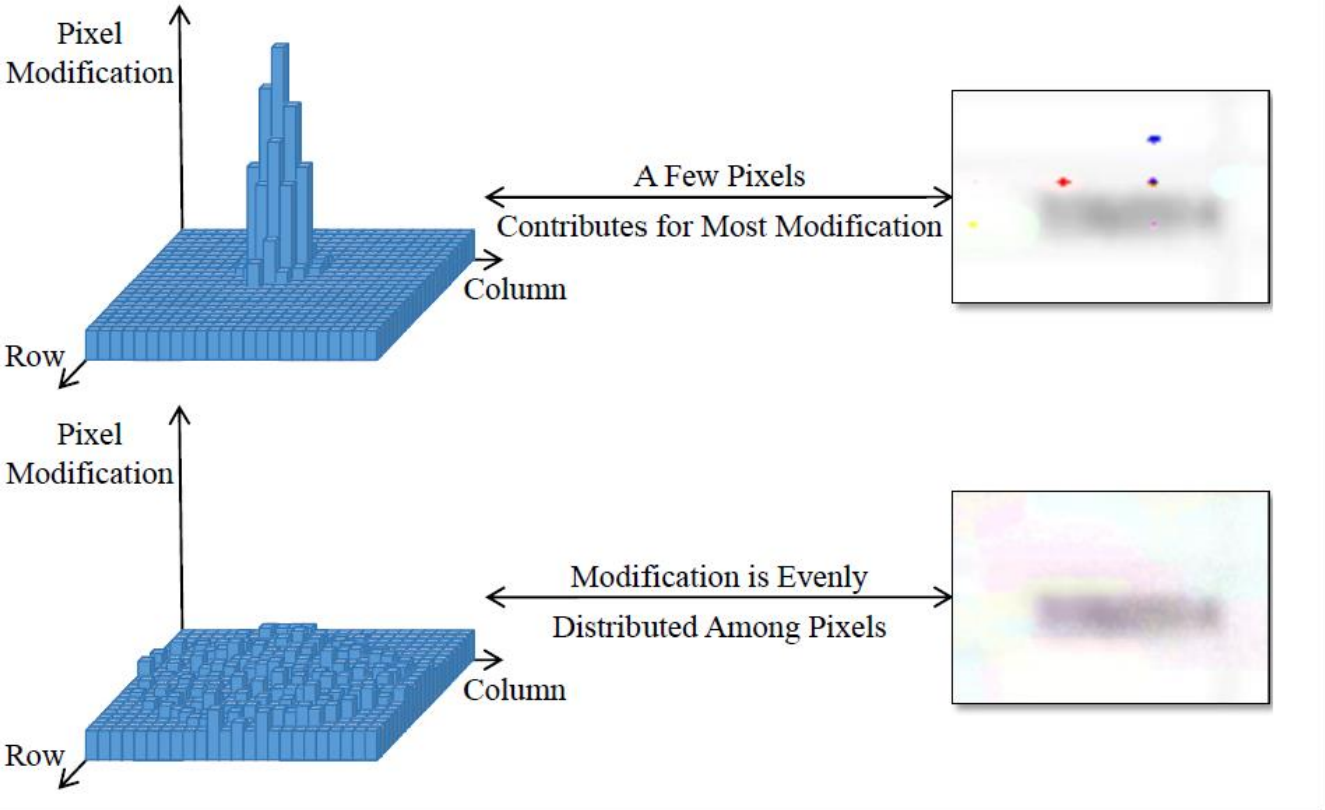


$|x_{adv} - x_{surrounding}|$

$$\min Z = C_{origin} \cdot |x_{adv} - x_{origin}| + C_{surrounding} \cdot |x_{adv} - x_{surrounding}|$$

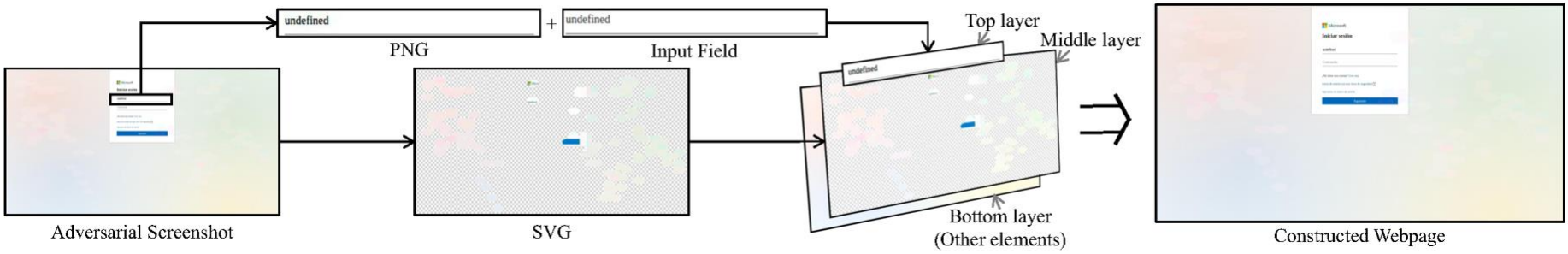
逆降采样——视觉可用性约束

- “**摊平**”像素值改变的影响，使得扰动尽可能均匀分布，确保视觉可用性。



构建攻击网页

- 逆降采样得到的图像转换为全尺寸的SVG图片，SVG中仅保留被修改的像素点，其余部分被挖空（透明）。
- 扰动像素点与输入交互元素重叠时，相应的扰动像素部分从SVG中挖出来，保存为PNG图片。
- 利用CSS的**z-index**将网页分层：
  - 输入交互元素放置在最上层，扰动所对应的PNG图片置为其背景——不遮盖用户输入；
  - SVG图片放在中间层，并使其能被“**点穿**”——不影响鼠标点击；
  - 网页原有的其他元素置于最底层。
- 所获得的网页在引入扰动像素的同时，还能保持功能性不变。



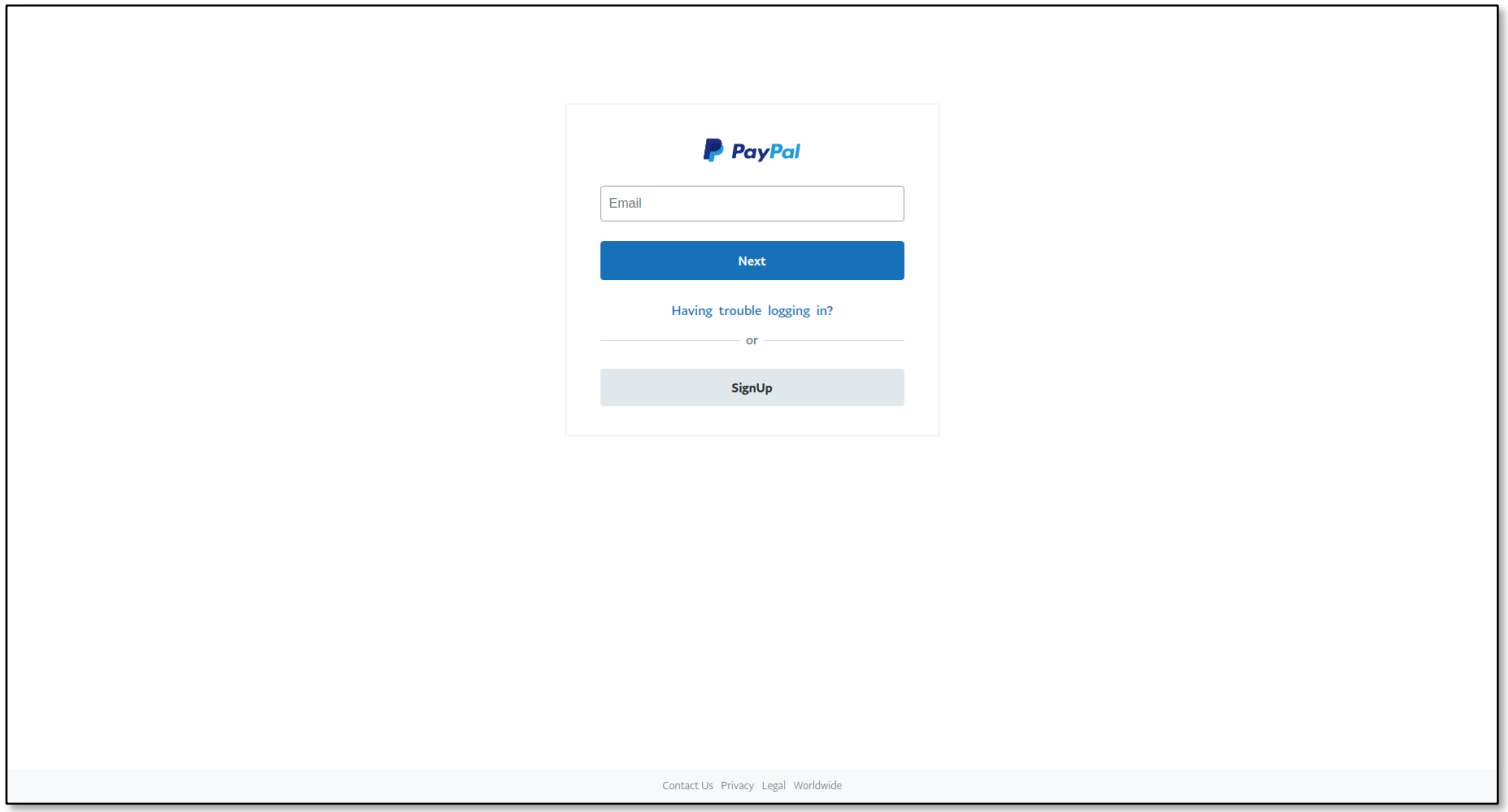
实验

- **实验数据：**135个真实钓鱼网页（收集自PhishBank和OpenPhish）；
- **实验对象：**最新版Chrome/Chromium。

样本数量	攻击成功数量	攻击成功率	平均耗时 （对抗样本生成+逆降采样）	平均显著性相似度 （以AUC衡量）
135	135	**100%**	10.7分钟	0.959

结论：我们的方法能够成功完全绕过Google新一代钓鱼网页检测分类器，并确保了攻击网页的视觉可用性。

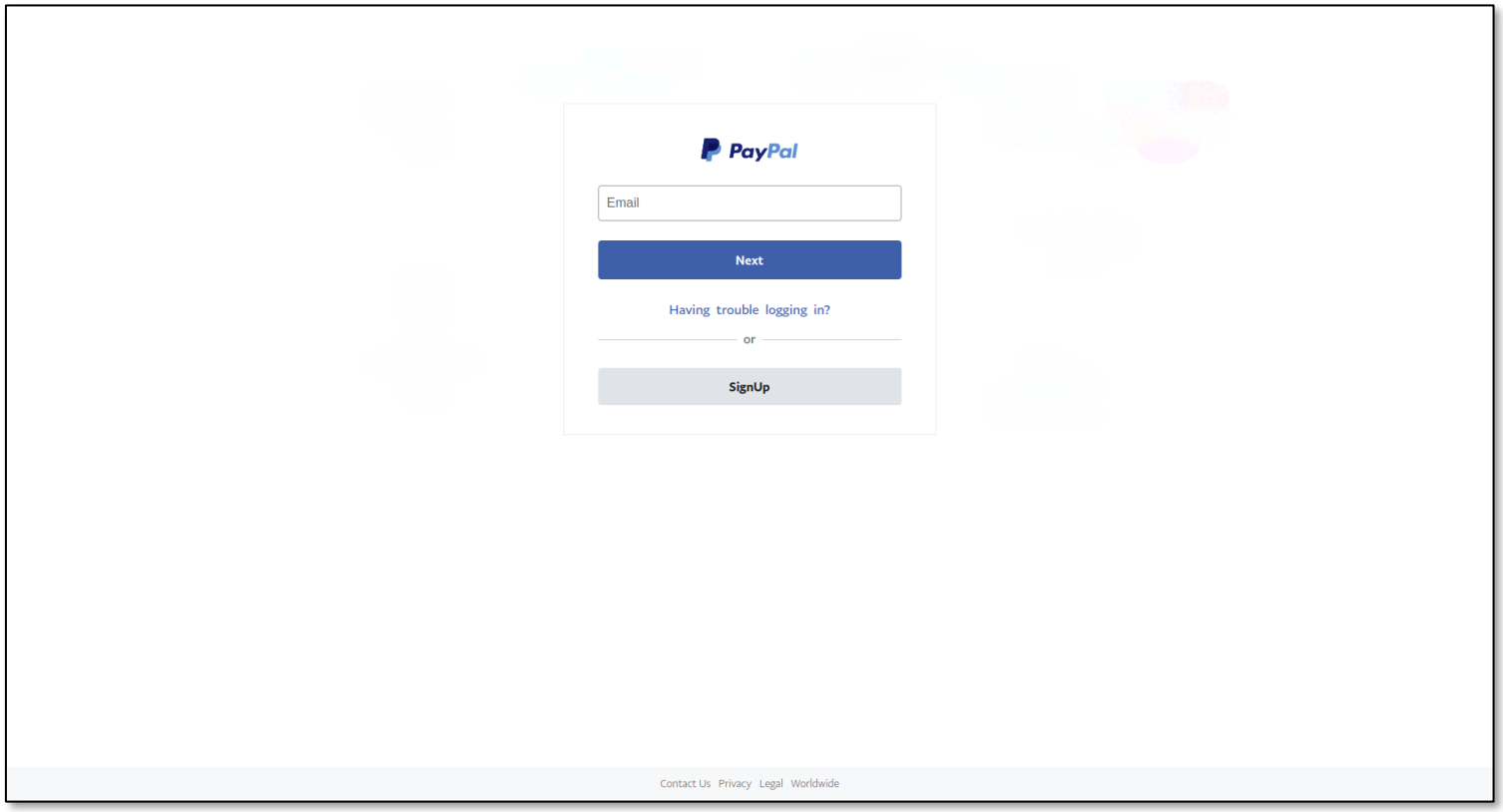
案例 – 原始钓鱼网页



分类分值:

0.91（钓鱼网页）

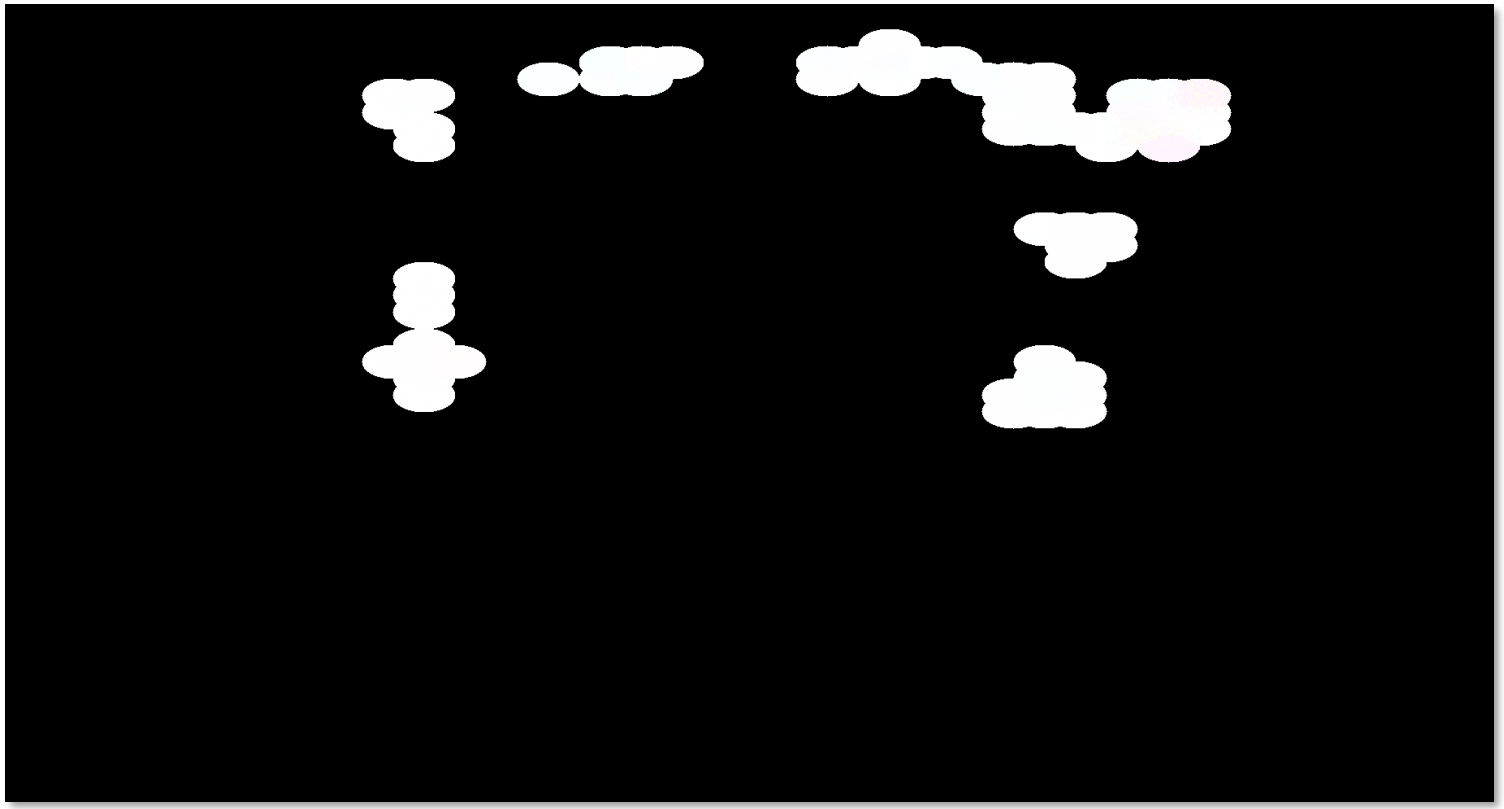
案例 – 对抗攻击网页



分类分值:

0.20（正常网页）

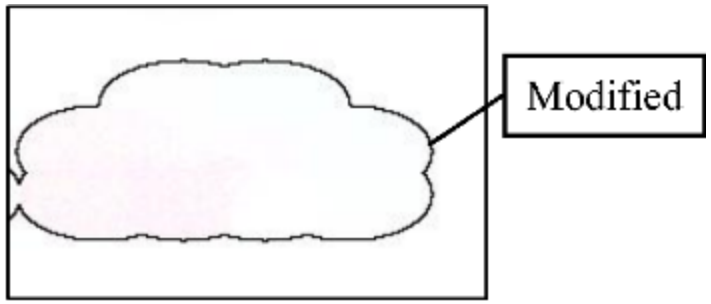
案例 – 扰动像素位置



案例 ——修改细节

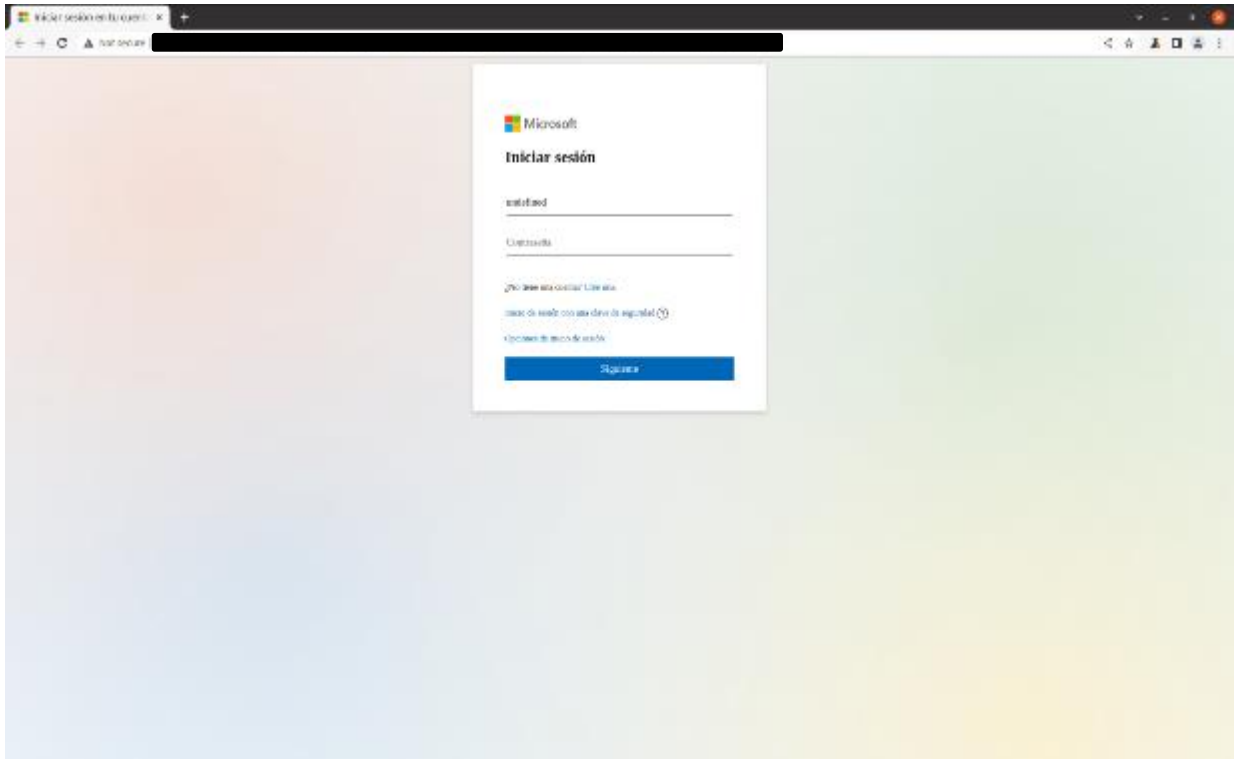
- 较为平滑的修改，使扰动不会显得太突兀，对网页浏览者而言，可能更像是屏幕上的污渍，网页能够较好地保持视觉可用性。

Original Part

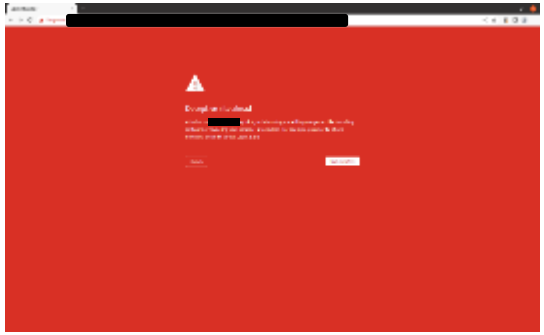


Modified Part

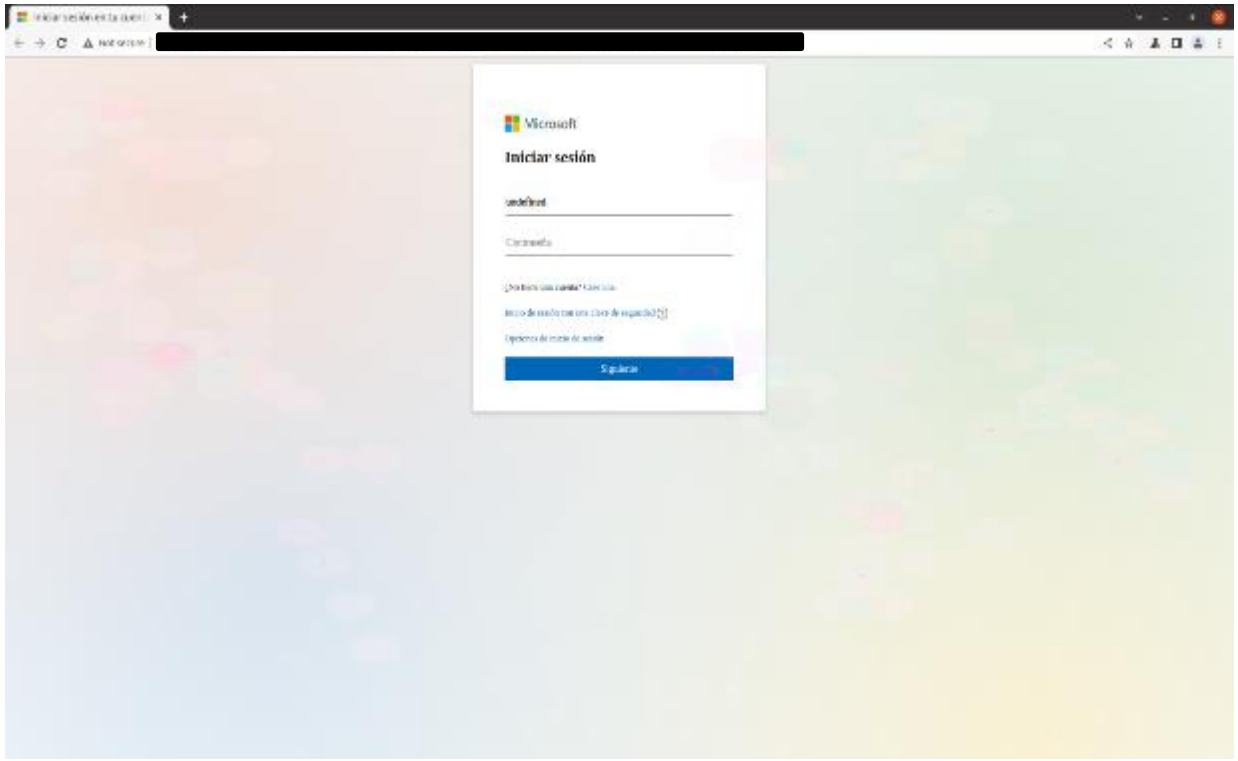
More Case—1



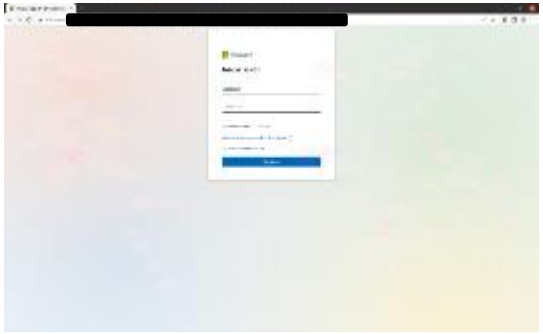
Blocked !



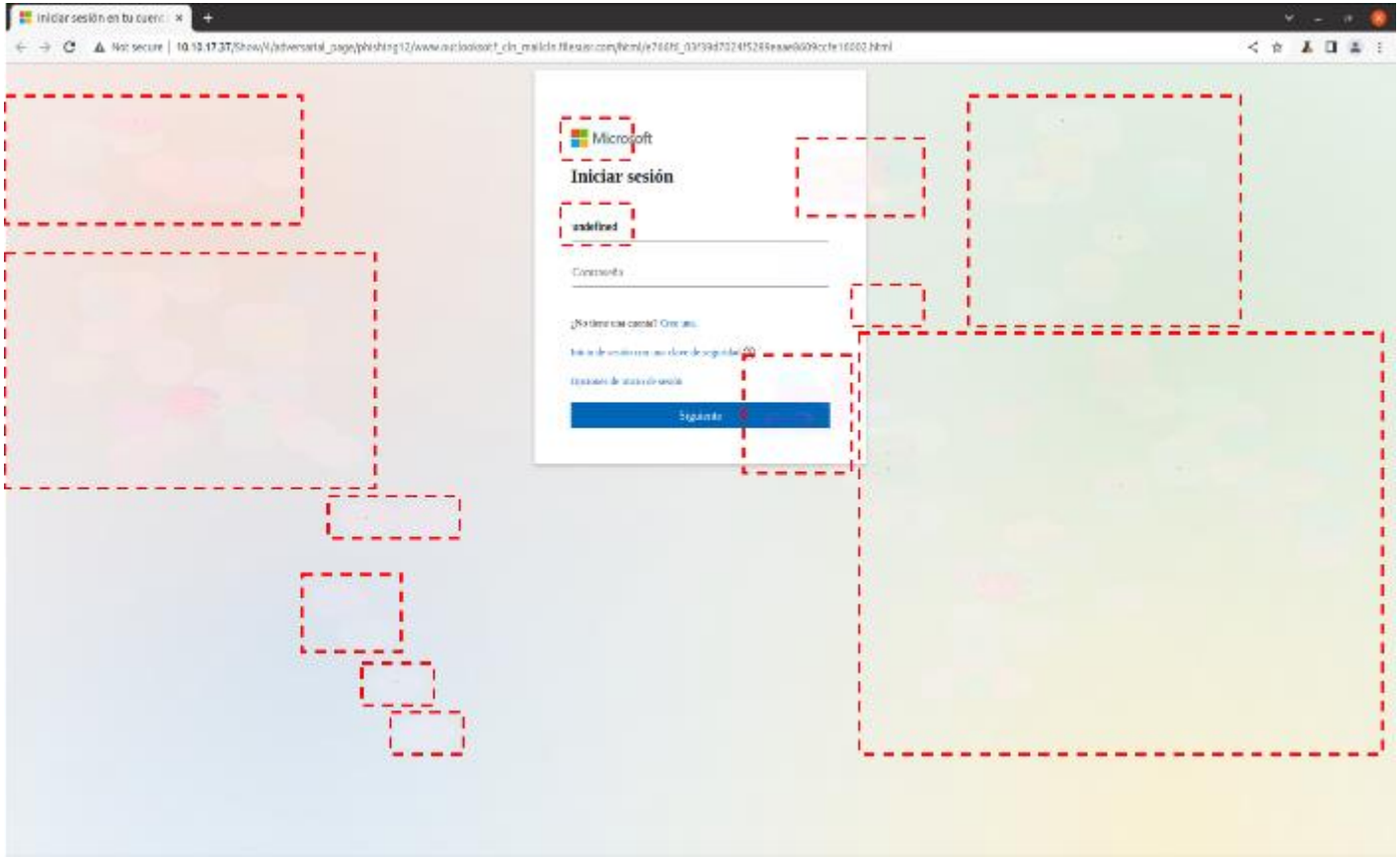
More Case—1



NOT blocked.

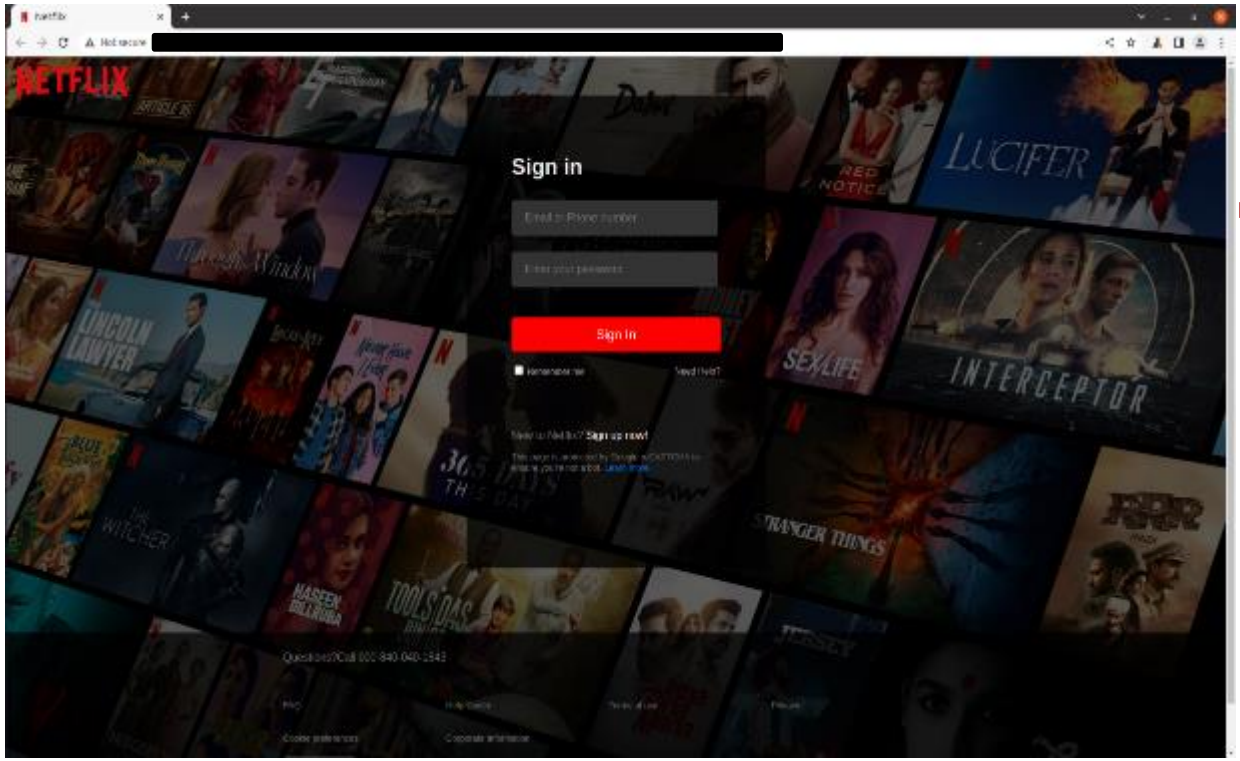


More Case—1

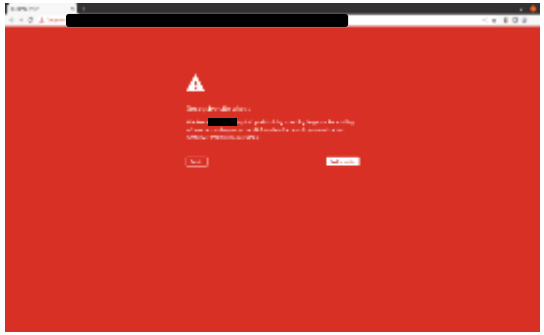


扰动区域

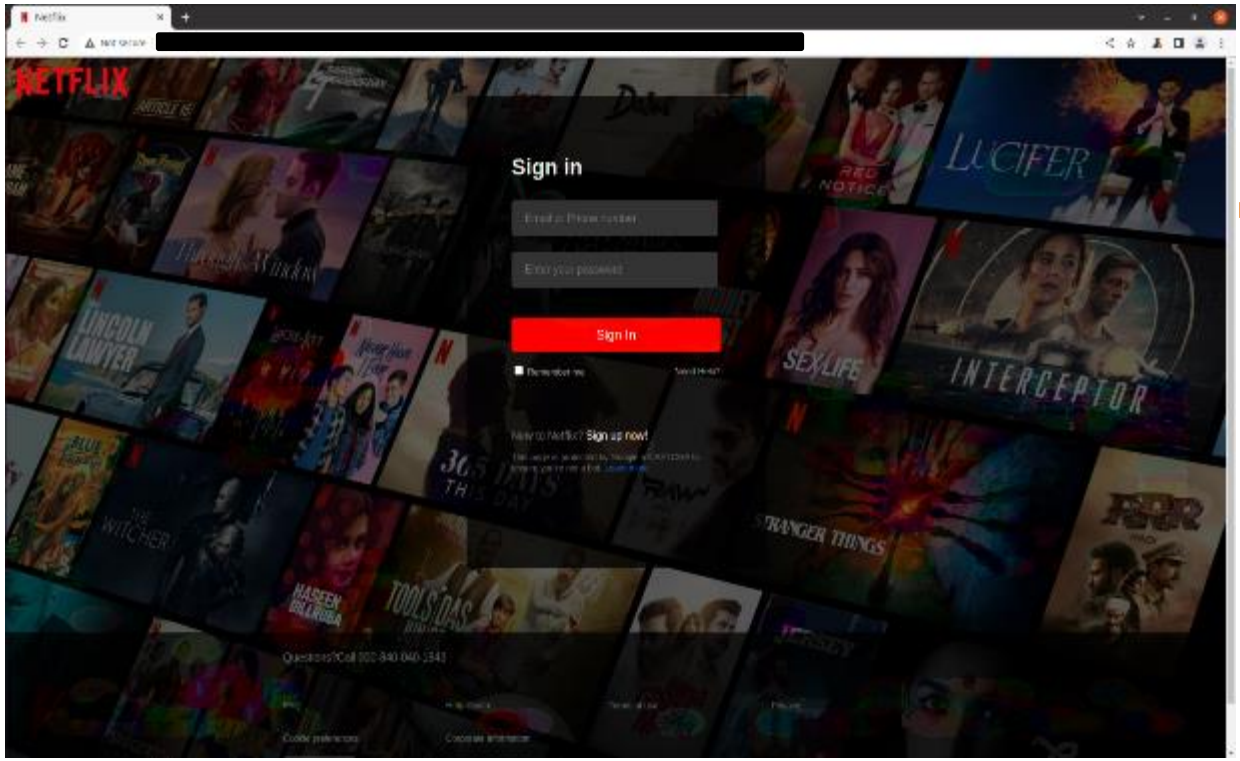
More Case—2



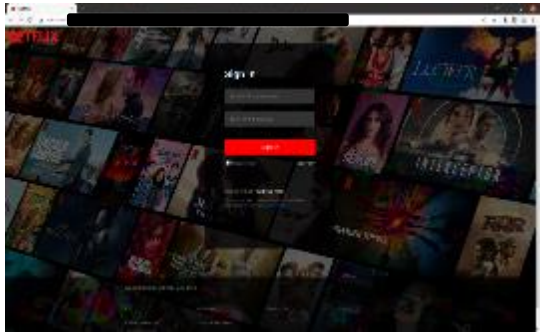
Blocked !



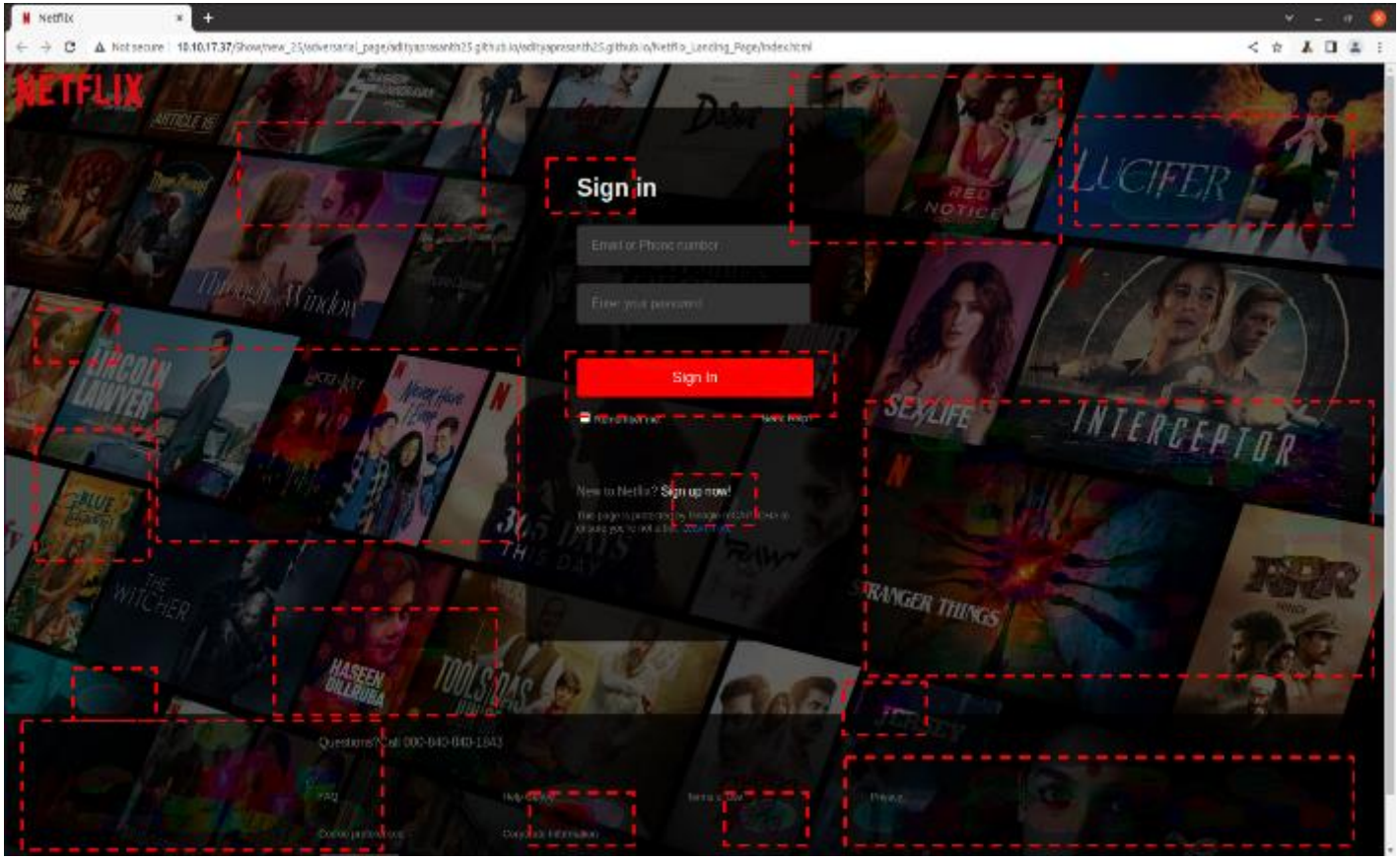
More Case—2



NOT blocked.

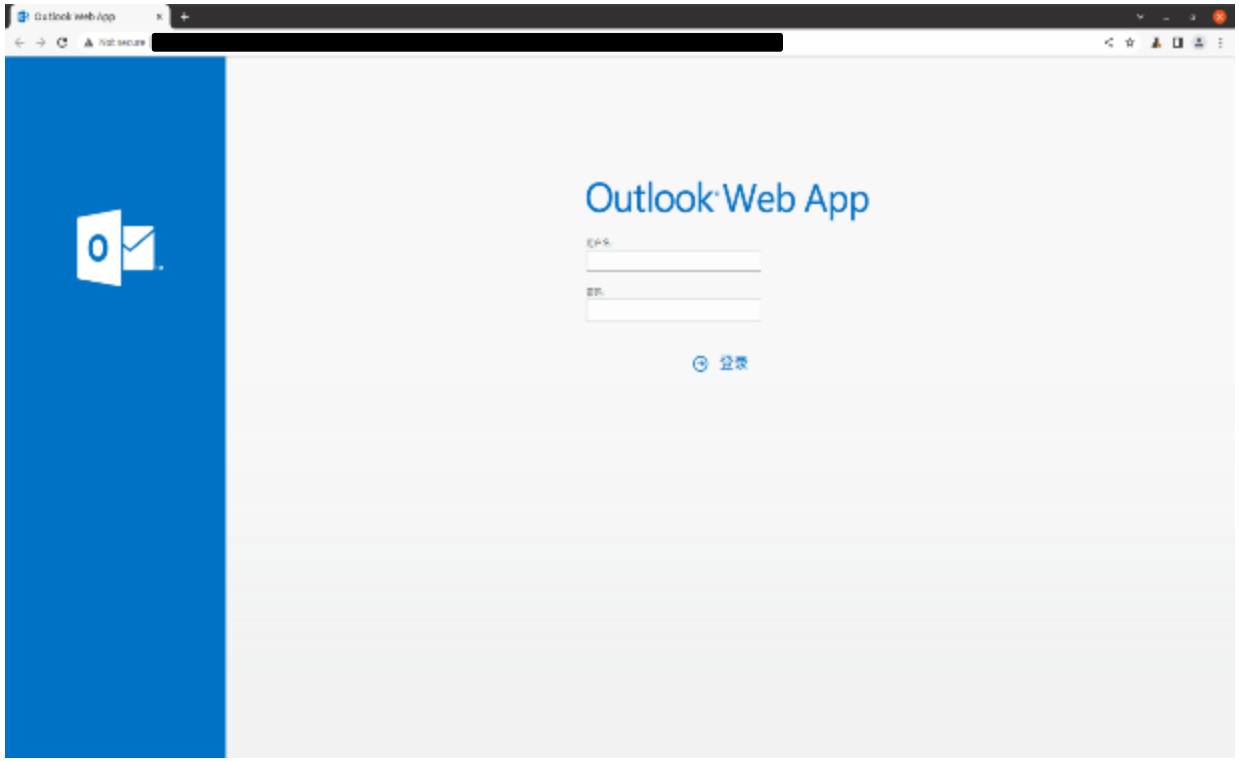


More Case—2

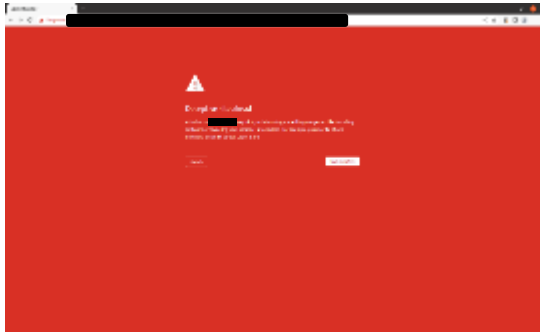


扰动区域

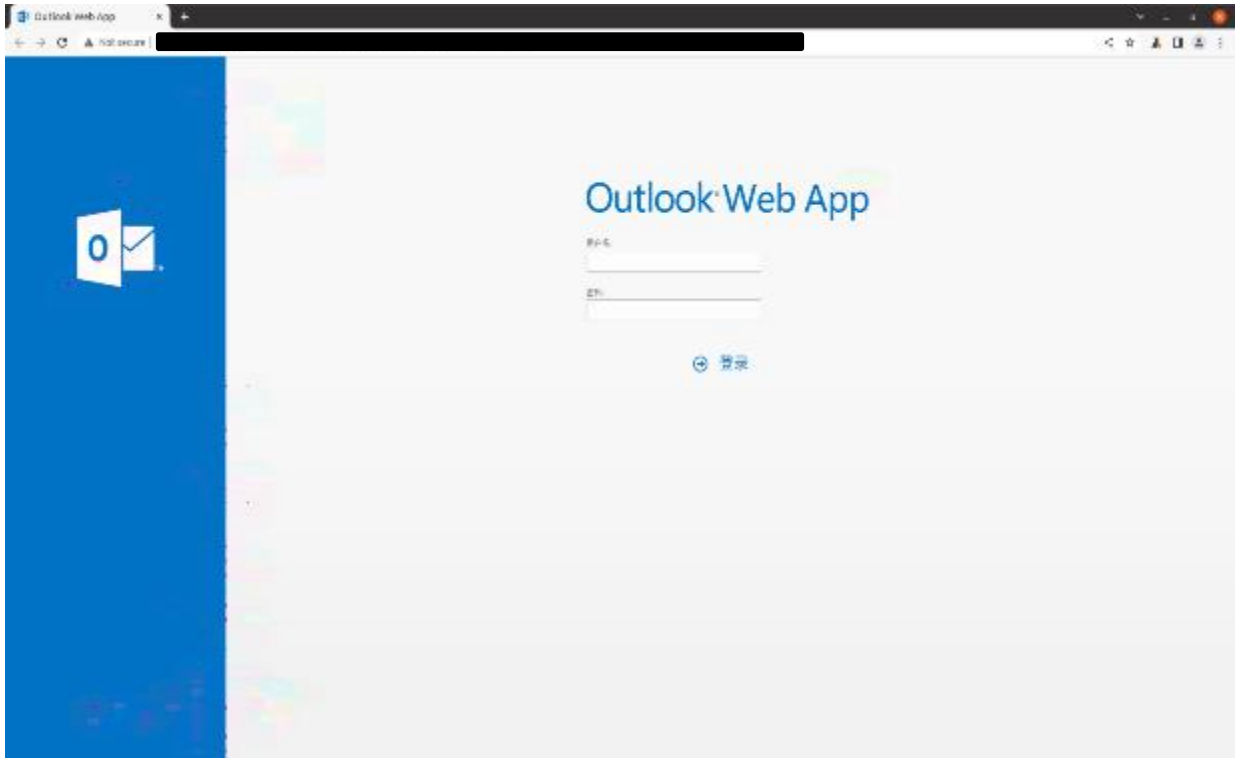
More Case—3



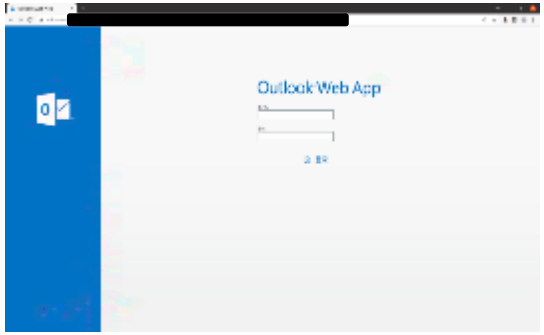
Blocked !



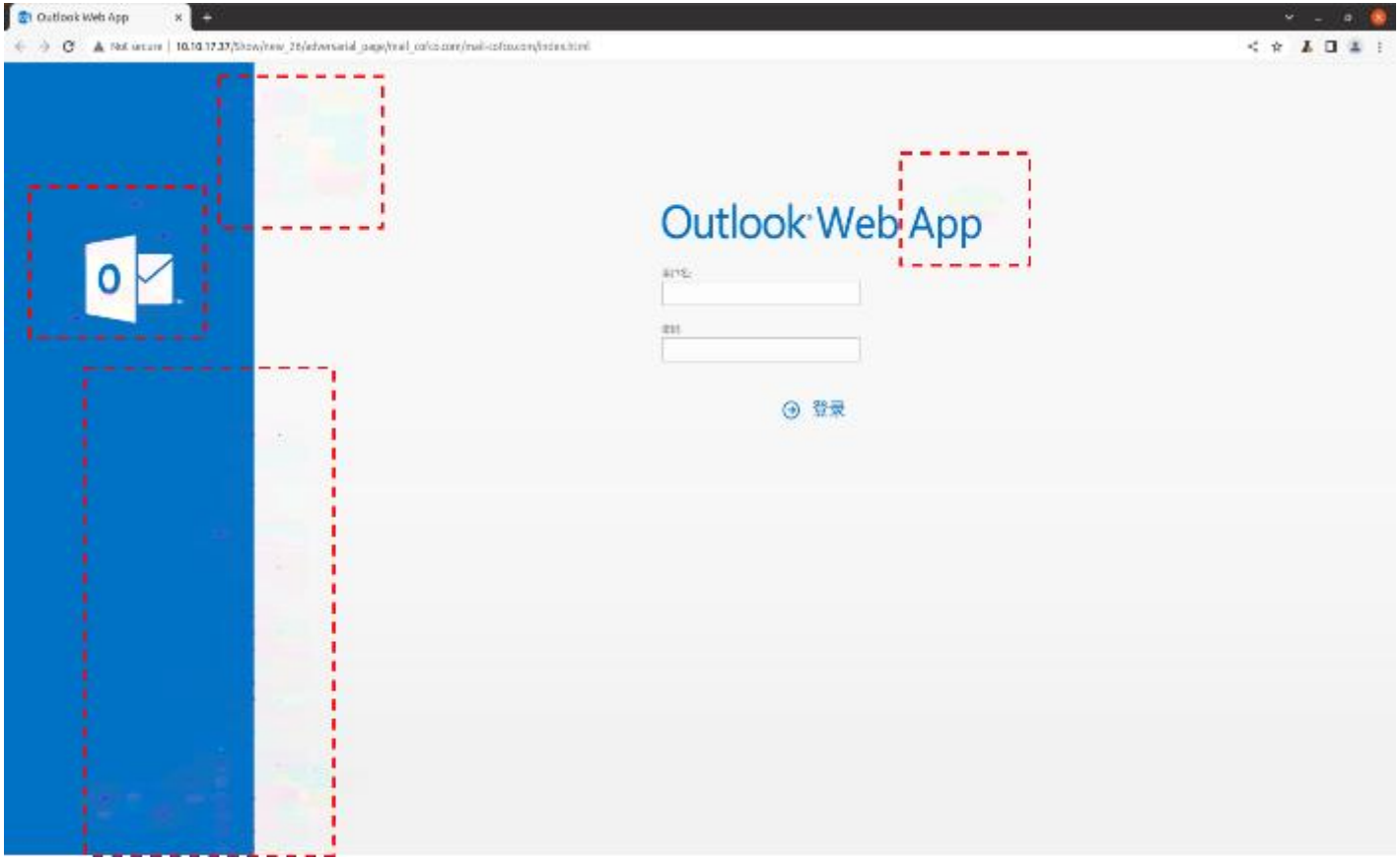
More Case—3



NOT blocked.



More Case—3



扰动区域