CSE P564: Computer Security and Privacy

Anonymity, Final Project, Usability

Autumn 2024

David Kohlbrenner

dkohlbre@cs

UW Instruction Team: David Kohlbrenner, Yoshi Kohno, Franziska Roesner. Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials

Paper discussion

Of Passwords and People: Measuring the Effect of Password-Composition Policies

Discuss!

- This study is from 2011, do you think anything would have changed if you ran it in 2024?
- Why were things like digit or symbol distribution not more uniform?
- The authors state that the entropy is an underestimate. Do you feel like the cracking is an underestimate or an overestimate or…?

Final Project Summary

Root-Cause Analysis (RCA)

- Suppose you work on the security team at a company
- You receive a report of a vulnerability in the wild, including a working proof-of-concept exploit
- What do you do now?

Root-Cause Analysis (RCA)

- Basically debugging, but you didn’t generate the input

- Consider:
  - What is different between a “normal” interaction and the exploit?
  - What part(s) of the program are relevant to that interaction
    - Add logging/debugging here! But consider that it might affect the exploit…
  - What specifically happened that was “unusual”?
  - Develop theories about what is happening
  - Test your theories!

The Goals for RCAs

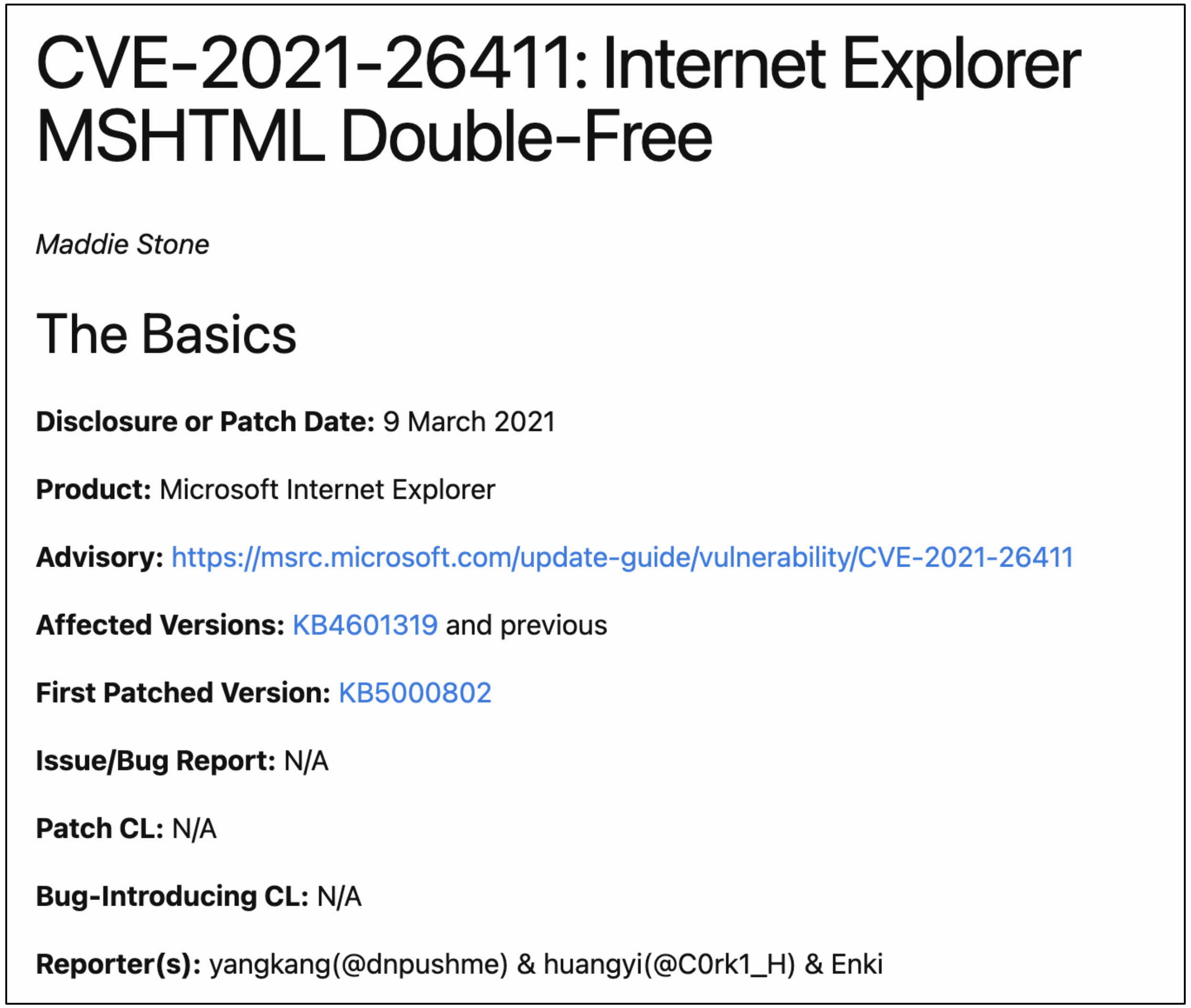
- Identify what the exploit accomplishes
- Identify the major steps the exploit takes
- Find the specific code components (if any exist) that are responsible
  - Aka: the vulnerability
  - Consider that an exploit might leverage *missing* features!
- Find “nearby” bugs
  - i.e., if you fix the most-responsible line of code, is it still vulnerable?
- Plan out a patch

Project 0 (p0) RCAs

- Google Project Zero (aka p0) is the premiere publicly-disclosing bug hunting team
- They produce detailed writeups of many bugs, and work with Google's Threat Analysis Group (aka TAG) to produce RCAs of in-the-wild bugs.
- You should go read some p0 RCAs! https://googleprojectzero.github.io/0days-in-the-wild/rca.html

https://googleprojectzero.github.io/0days-in-the-wild/0day-RCAs/2021/CVE-2021-26411.html

Sample RCA



CVE-2021-26411: Internet Explorer MSHTML Double-Free

Maddie Stone

The Basics

Disclosure or Patch Date: 9 March 2021

Product: Microsoft Internet Explorer

Advisory: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-26411

Affected Versions: KB4601319 and previous

First Patched Version: KB5000802

Issue/Bug Report: N/A

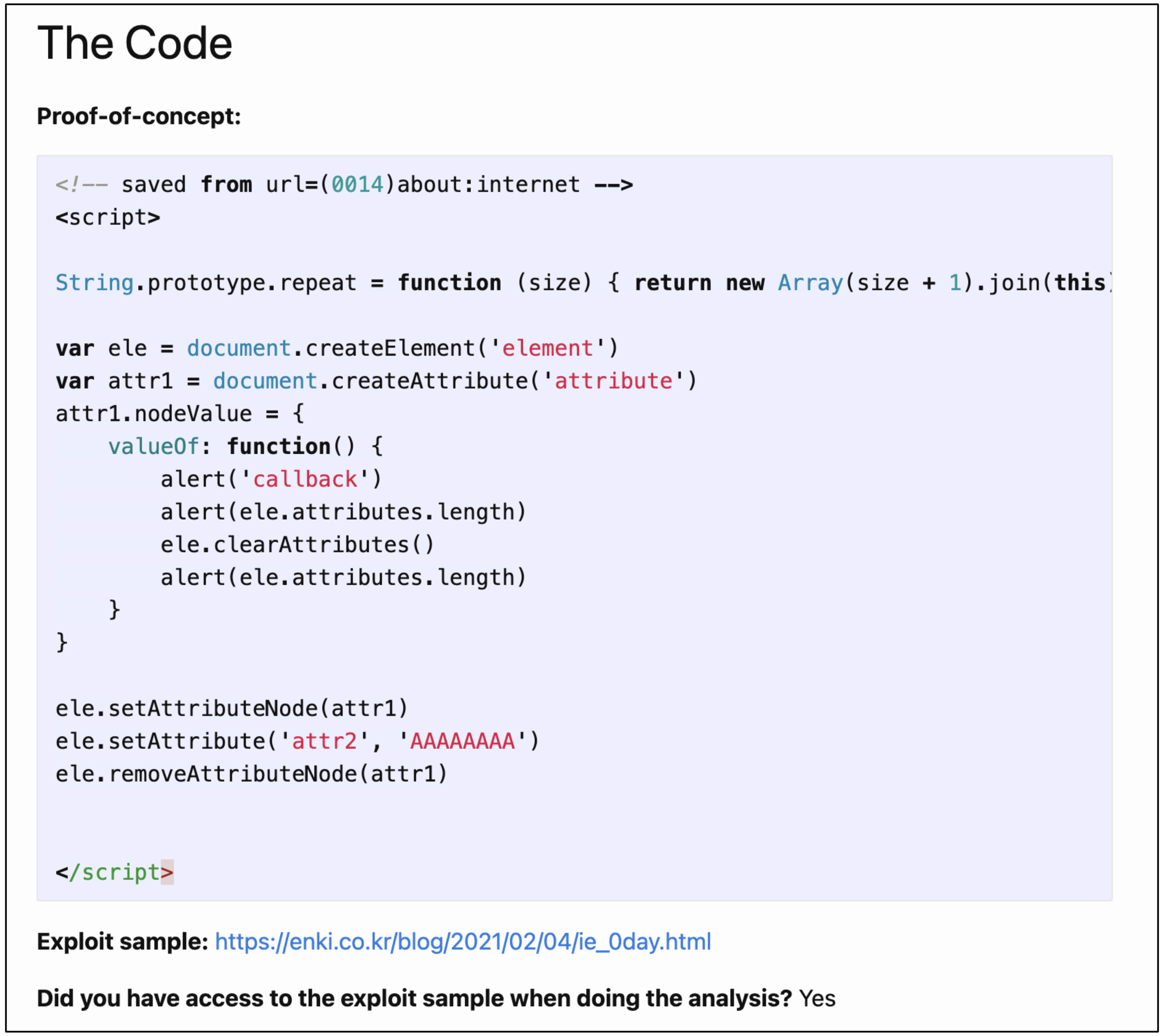
Patch CL: N/A

Bug-Introducing CL: N/A

Reporter(s): yangkang(@dnpushme) & huangyi(@C0rk1_H) & Enki

https://googleprojectzero.github.io/0days-in-the-wild/0day-RCAs/2021/CVE-2021-26411.html

Sample RCA



The Code

Proof-of-concept:

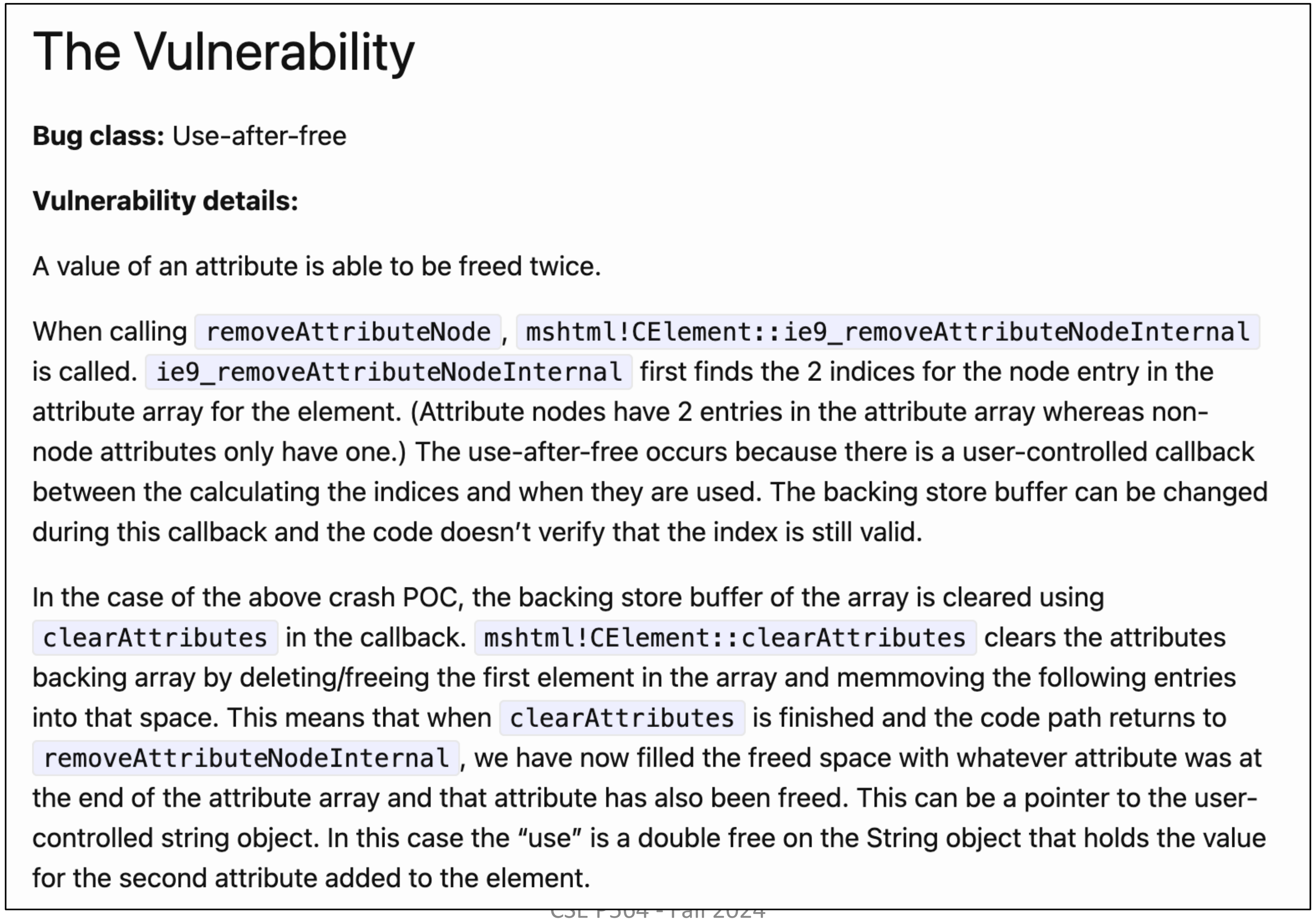
```
<!-- saved from url=(0014)about:internet -->
<script>

String.prototype.repeat = function (size) { return new Array(size + 1).join(this

var ele = document.createElement('element')
var attr1 = document.createAttribute('attribute')
attr1.nodeValue = {
    valueOf: function() {
        alert('callback')
        alert(ele.attributes.length)
        ele.clearAttributes()
        alert(ele.attributes.length)
    }
}

ele.setAttributeNode(attr1)
ele.setAttribute('attr2', 'AAAAAAAA')
ele.removeAttributeNode(attr1)


</script>
```

Exploit sample: https://enki.co.kr/blog/2021/02/04/ie_0day.html

Did you have access to the exploit sample when doing the analysis? Yes

https://googleprojectzero.github.io/0days-in-the-wild/0day-RCAs/2021/CVE-2021-26411.html

Sample RCA



The Vulnerability

Bug class: Use-after-free

Vulnerability details:

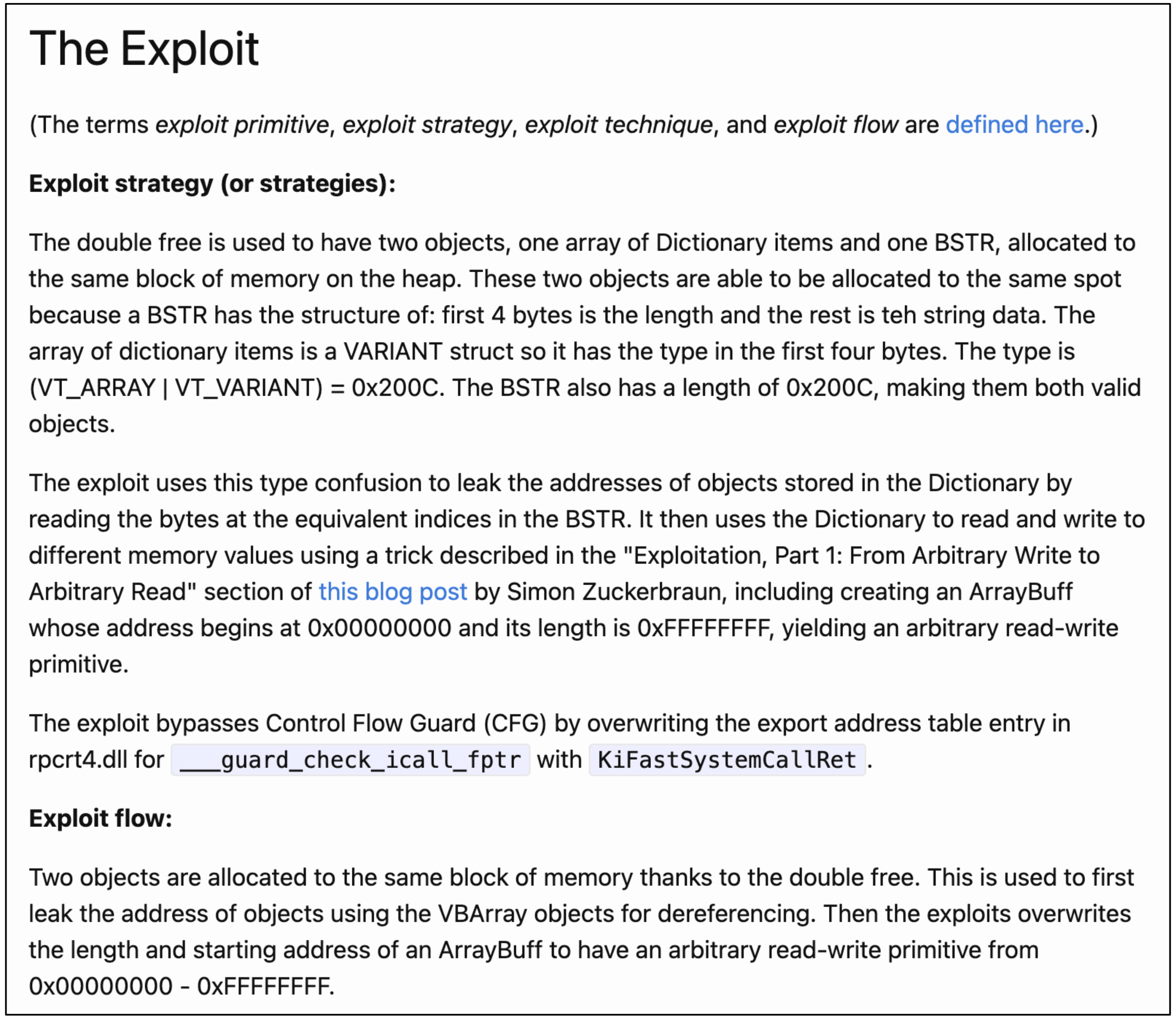
A value of an attribute is able to be freed twice.

When calling `removeAttributeNode`, `mshtml!CElement::ie9_removeAttributeNodeInternal` is called. `ie9_removeAttributeNodeInternal` first finds the 2 indices for the node entry in the attribute array for the element. (Attribute nodes have 2 entries in the attribute array whereas non-node attributes only have one.) The use-after-free occurs because there is a user-controlled callback between the calculating the indices and when they are used. The backing store buffer can be changed during this callback and the code doesn't verify that the index is still valid.

In the case of the above crash POC, the backing store buffer of the array is cleared using `clearAttributes` in the callback. `mshtml!CElement::clearAttributes` clears the attributes backing array by deleting/freeing the first element in the array and memmoving the following entries into that space. This means that when `clearAttributes` is finished and the code path returns to `removeAttributeNodeInternal`, we have now filled the freed space with whatever attribute was at the end of the attribute array and that attribute has also been freed. This can be a pointer to the user-controlled string object. In this case the "use" is a double free on the String object that holds the value for the second attribute added to the element.

https://googleprojectzero.github.io/0days-in-the-wild/0day-RCAs/2021/CVE-2021-26411.html

Sample RCA



The Exploit

(The terms *exploit primitive*, *exploit strategy*, *exploit technique*, and *exploit flow* are defined here.)

Exploit strategy (or strategies):

The double free is used to have two objects, one array of Dictionary items and one BSTR, allocated to the same block of memory on the heap. These two objects are able to be allocated to the same spot because a BSTR has the structure of: first 4 bytes is the length and the rest is teh string data. The array of dictionary items is a VARIANT struct so it has the type in the first four bytes. The type is (VT_ARRAY | VT_VARIANT) = 0x200C. The BSTR also has a length of 0x200C, making them both valid objects.

The exploit uses this type confusion to leak the addresses of objects stored in the Dictionary by reading the bytes at the equivalent indices in the BSTR. It then uses the Dictionary to read and write to different memory values using a trick described in the "Exploitation, Part 1: From Arbitrary Write to Arbitrary Read" section of this blog post by Simon Zuckerbraun, including creating an ArrayBuff whose address begins at 0x00000000 and its length is 0xFFFFFFFF, yielding an arbitrary read-write primitive.

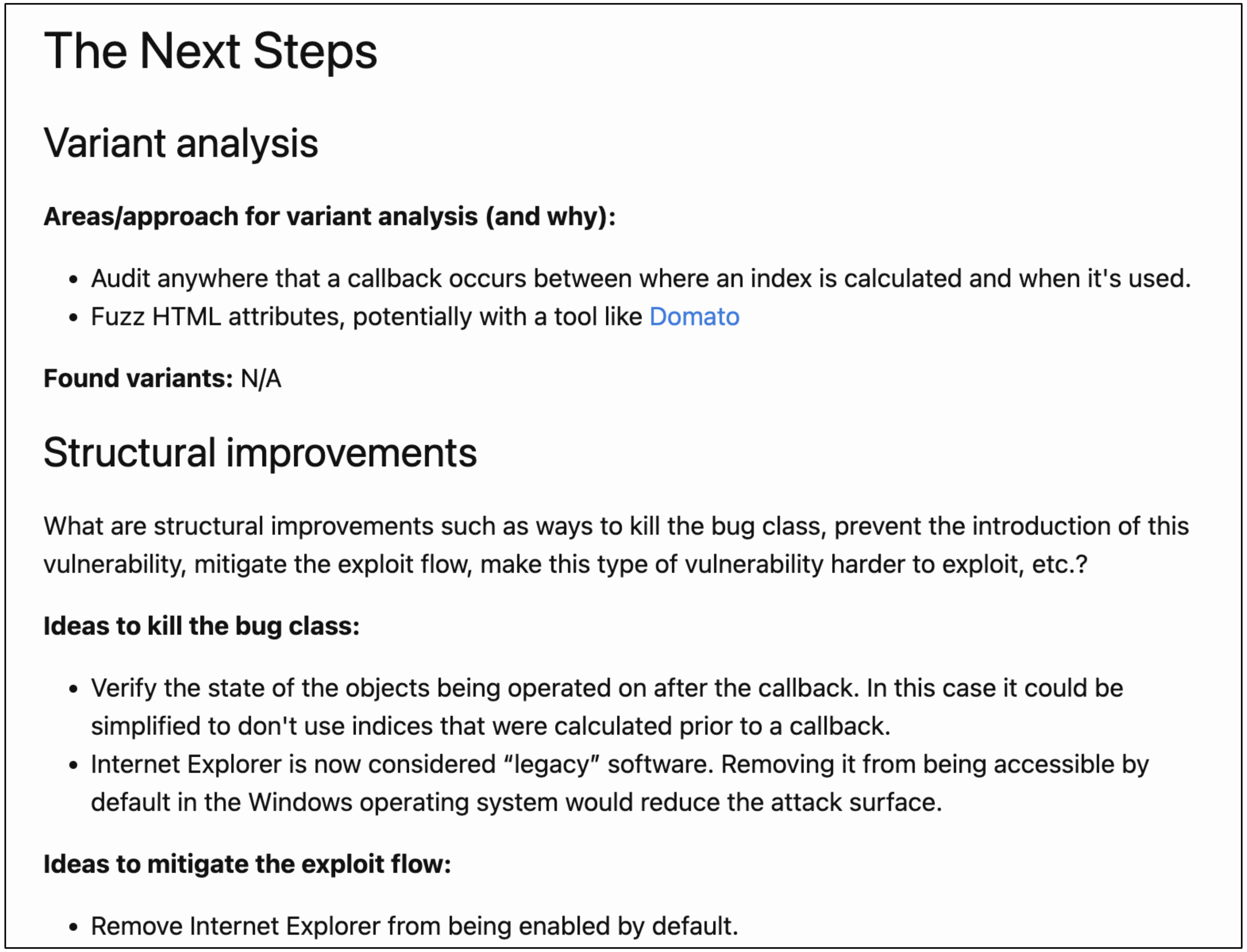
The exploit bypasses Control Flow Guard (CFG) by overwriting the export address table entry in rpcrt4.dll for `___guard_check_icall_fptr` with `KiFastSystemCallRet`.

Exploit flow:

Two objects are allocated to the same block of memory thanks to the double free. This is used to first leak the address of objects using the VBArray objects for dereferencing. Then the exploits overwrites the length and starting address of an ArrayBuff to have an arbitrary read-write primitive from 0x00000000 - 0xFFFFFFFF.

https://googleprojectzero.github.io/0days-in-the-wild/0day-RCAs/2021/CVE-2021-26411.html

Sample RCA



The Next Steps

Variant analysis

Areas/approach for variant analysis (and why):

- Audit anywhere that a callback occurs between where an index is calculated and when it's used.
- Fuzz HTML attributes, potentially with a tool like Domato

Found variants: N/A

Structural improvements

What are structural improvements such as ways to kill the bug class, prevent the introduction of this vulnerability, mitigate the exploit flow, make this type of vulnerability harder to exploit, etc.?

Ideas to kill the bug class:

- Verify the state of the objects being operated on after the callback. In this case it could be simplified to don't use indices that were calculated prior to a callback.
- Internet Explorer is now considered "legacy" software. Removing it from being accessible by default in the Windows operating system would reduce the attack surface.

Ideas to mitigate the exploit flow:

- Remove Internet Explorer from being enabled by default.

Patch Writing Goals

- Break the *specific* exploit strategy the exploit uses
- Break *similar* exploit strategies
  - Consider how XSS filtering worked in Lab 2! (… Not great!)
- Minimize breaking explicit features of the program
- Minimize breaking *implicit* features of the program

Public Bug Finding: Terminology

- "Zero Days" – 0 days (aka "o-days")
  - Refers to a bug that is made publicly known at the same time as the vendor is told
  - The vendor has had '0 days' of lead time to fix it
- CVE Number
  - Common Vulnerabilities and Exposures
  - E.g. CVE-2022-4135
- CWE
  - Common Weakness Enumeration
  - Standardized list of bug types
- CVSS
  - Common Vulnerability Scoring System
  - Very limited utility, scores barely correlated with impact

Public Bug Finding: Disclosure

- At some point, the vendor finds out about the bug
  - Either publicly revealed by finder (an 0-day)
  - Internally found by code auditing
  - Found being used in-the-wild
- If *you* find the bug:
  - When do you disclose?
  - How do you disclose?
  - “Full-disclosure” vs “Coordinated disclosure” vs “Responsible disclosure”
- Bug bounty programs offer incentives to disclose
  - But at a cost: you usually have to sign NDAs

Final Project Learning Goals

- Combine lessons/skills from the quarter:
  - Identifying and understanding (and fixing) vulnerabilities
  - Debugging and execution tracing (e.g., with gdb)
  - Software and web security concepts
  - Clear technical communication
- You'll gain experience in:
  - **Root-causing** a security bug similar to ones seen in class
  - **Writing patches** for security bugs similar to ones seen in class
  - **Making sense of a moderate codebase** (~1100 or fewer loc)
- This lab is a more realistic depiction of what working in security industry on the defender side in "real life" might be like

Final Project Components

- Part A
  - We will give you the RCA for an exploit, and **you have to write the patch**
- Part B
  - We will give you an exploit, and **you have to write the RCA**
  - (You can choose one of two exploits. You can do one additional for extra credit.)
- Part C
  - **You have to write the patch** for Part B's exploit

tinyserv – a tiny, bad, HTTP server

- ~1000 lines of C code
- Moderately well commented
- Quite buggy ☺
- You can interact with it via command line tools or a web browser

Major Features

- “admin” login
  - Sets a randomized password on server start
  - Successful login sets a cookie that lets admins access admin.txt
  - admin.txt contains a log of requests received so far
  - (Our exploits work by demonstrating they can access admin.txt)
- Dynamic content fills
  - Some pages have dynamic content (notably 404s) that gets filled at request
- Response caching
  - Pages are cached in a hashtable on first send
  - Future responses will check the hashtable first

How Should You Start?

- To run it, inside target/tinyserv: `./tinyserv ./files`
- In browser (from anywhere), visit:
  - http://umnak.cs.washington.edu:YOUR-PORT-NUMBER
  - Find the port number in the `lab3_port` file, and your group's secret in the `lab3_group_secret` file

- (FYI, to minimize risk, the server will kill itself after 3 hours if you leave it running)

Quick Demo

- Notes from demo:
  - “make” inside target/tinyserv to (re)compile tinyserv
  - curl: a tool that generates an HTTP request, used in sploits
  - “./handin.sh” to create a diff after you’ve created a patch

RCA Strategies

- Read through the server code (see main.c to start)
  - You don’t have to understand everything!
- Read through the sploit inputs and try to guess which parts of the tinyserv code might be related; start debugging there!
- Use gdb for debugging and execution tracing
  - `gdb –args ./tinyserv ./files`
  - `break [function name or line number]`
  - `run`
  - From another terminal window, you can now run the sploits
- (Maybe:) Modify main.c to test things out or add print statements

Other Final Project Notes

- There is an additional cookie: the lab group secret key
    - This is NOT part of the lab, it is there to prevent accidentally interacting with other groups' servers
- You also don't need to dig into the socket-related code

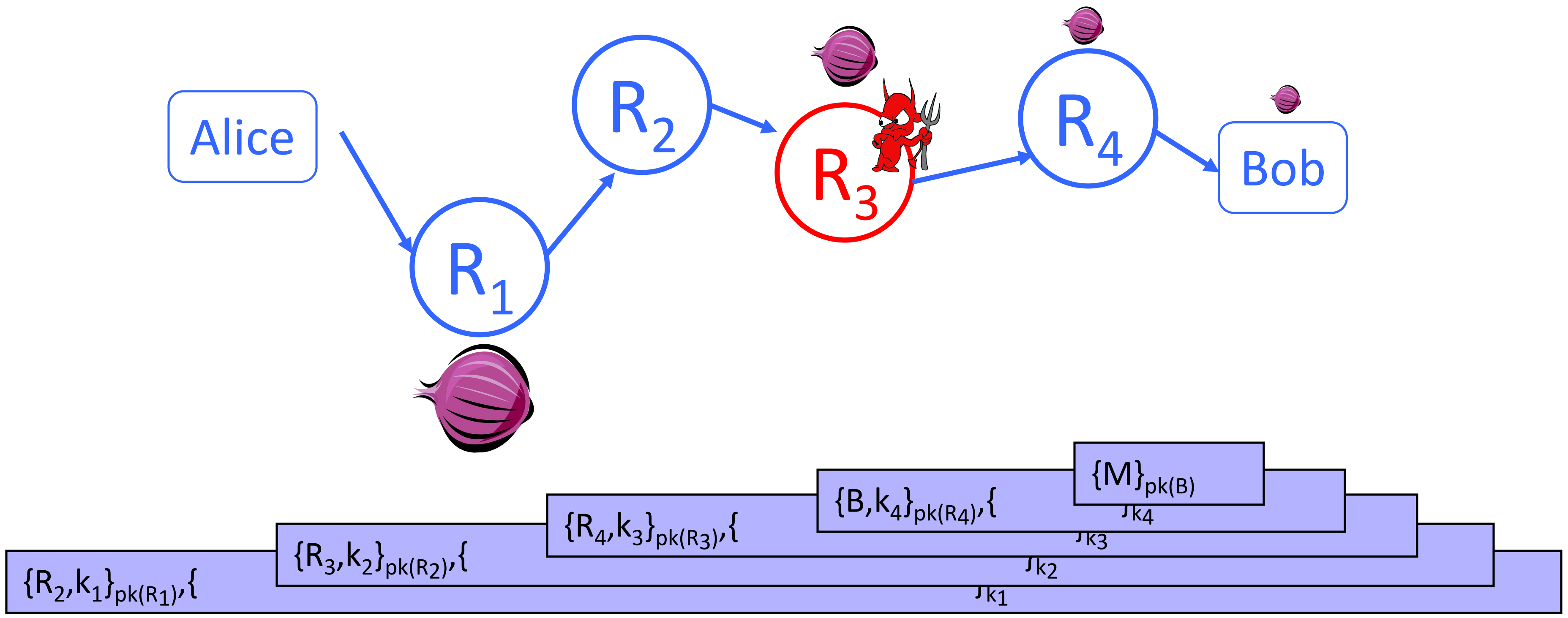
Turn-in (Group Submissions)

There are 5 Gradescope assignments:

- **Everyone submits to this one:**
  - Final Project Part A – Sploit1
- Submit to **ONE** of these, depending on which sploit you do:
  - Final Project Part B – Sploit3 Version
  - Final Project Part B – Sploit4 Version
- Submit to **ONE** of these, matching your part B:
  - Final Project Part C – Sploit3 Version
  - Final Project Part C – Sploit4 Version

TOR and Anonymity

Onion Routing



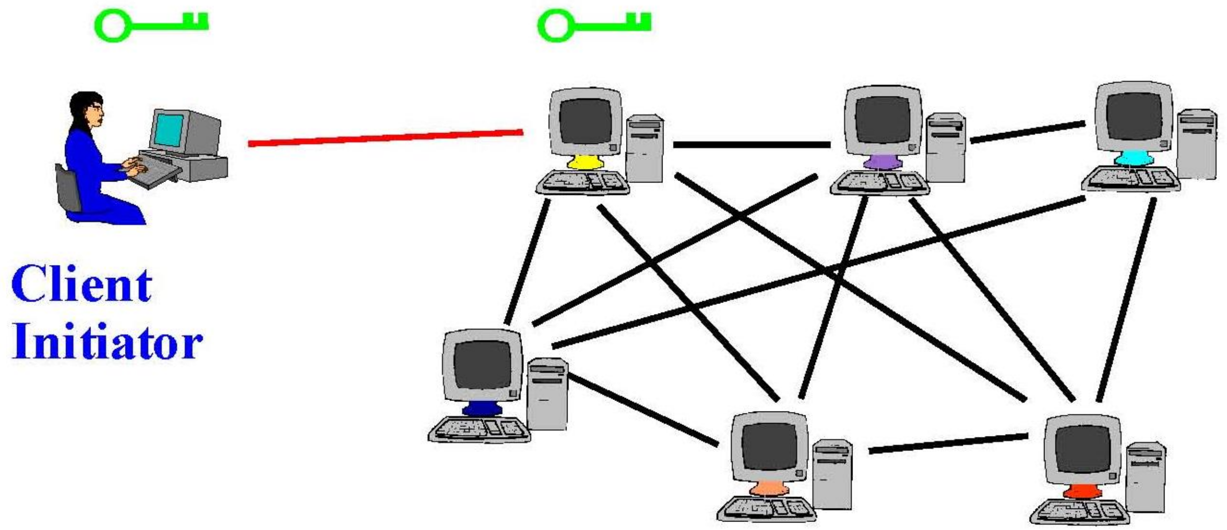
- Routing info for each link encrypted with router's public key
- Each router learns only the identity of the next router

Tor

- Second-generation onion routing network
  - http://tor.eff.org
  - Developed by Roger Dingledine, Nick Mathewson and Paul Syverson
  - Specifically designed for low-latency anonymous Internet communications
- Running since October 2003
- “Easy-to-use” client proxy
  - Freely available, can use it for anonymous browsing

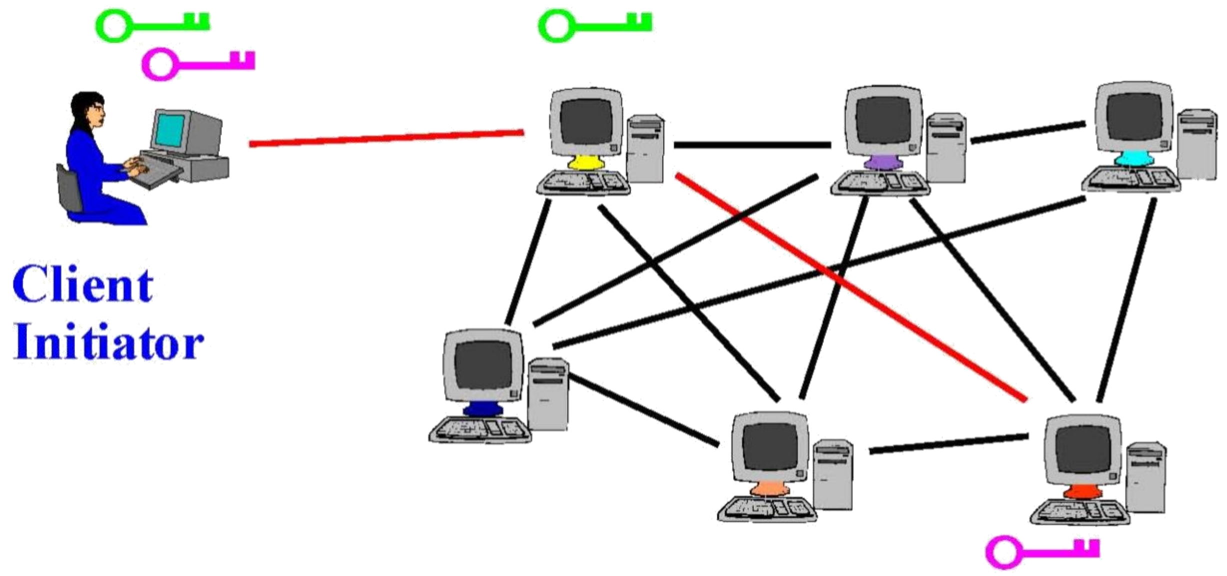
Tor Circuit Setup (1)

- Client proxy establishes a symmetric session key and circuit with Onion Router #1



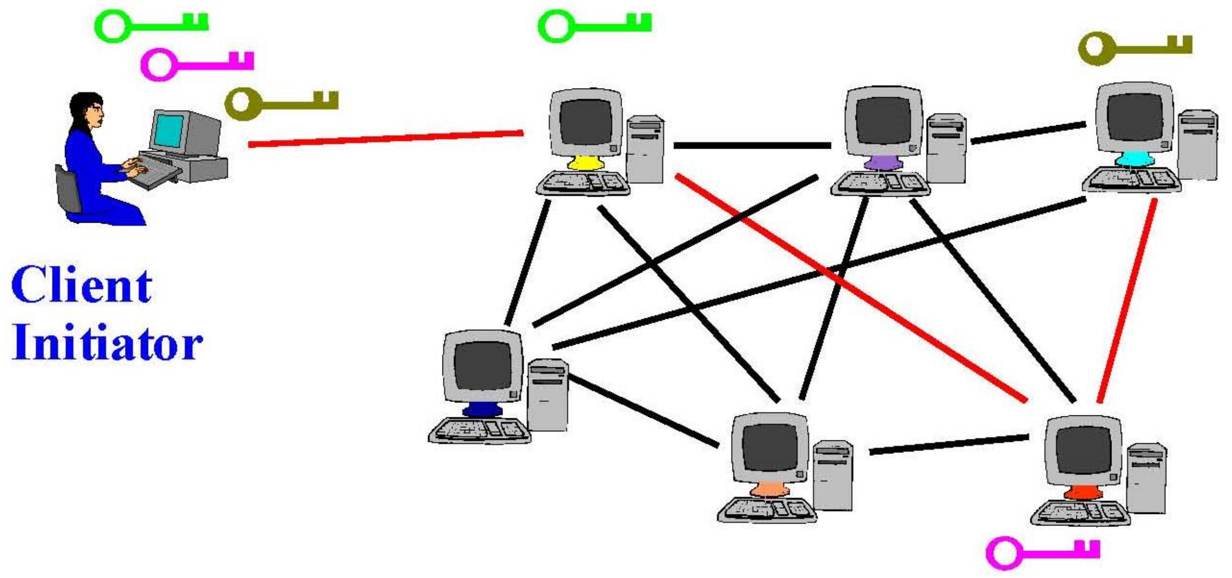
Tor Circuit Setup (2)

- Client proxy extends the circuit by establishing a symmetric session key with Onion Router #2
  - Tunnel through Onion Router #1



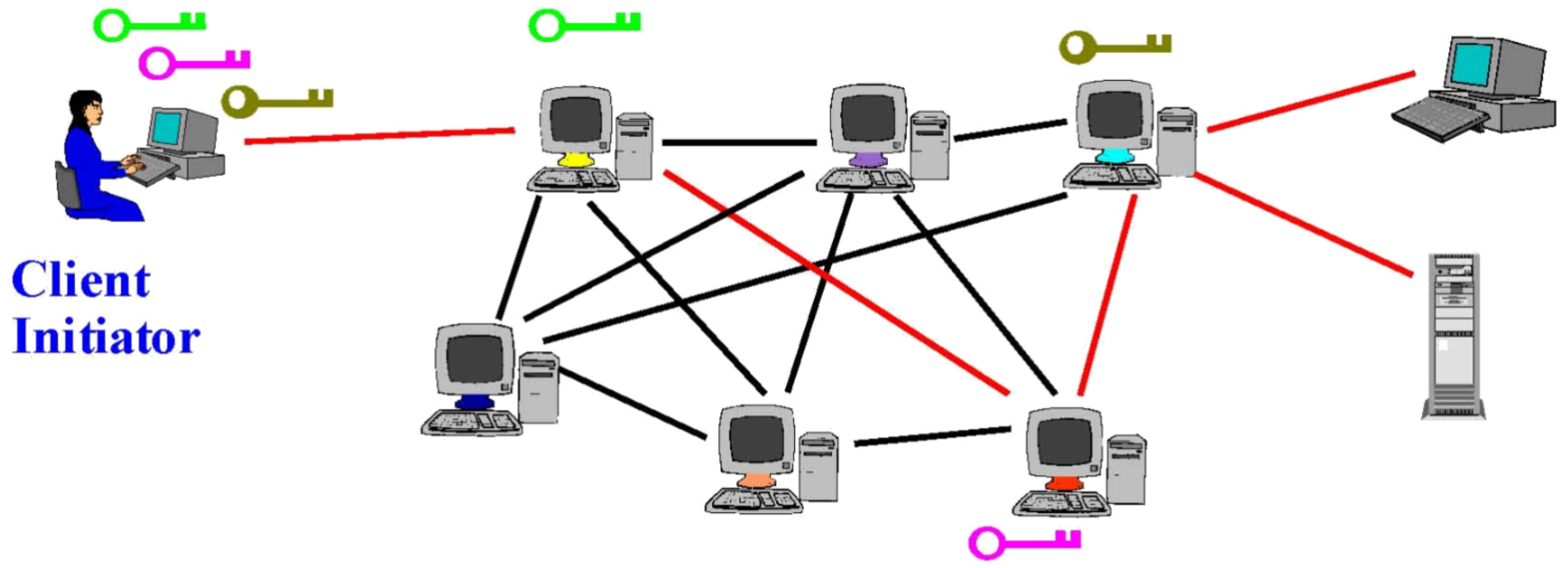
Tor Circuit Setup (3)

- Client proxy extends the circuit by establishing a symmetric session key with Onion Router #3
  - Tunnel through Onion Routers #1 and #2



Using a Tor Circuit

- Client applications connect and communicate over the established Tor circuit.



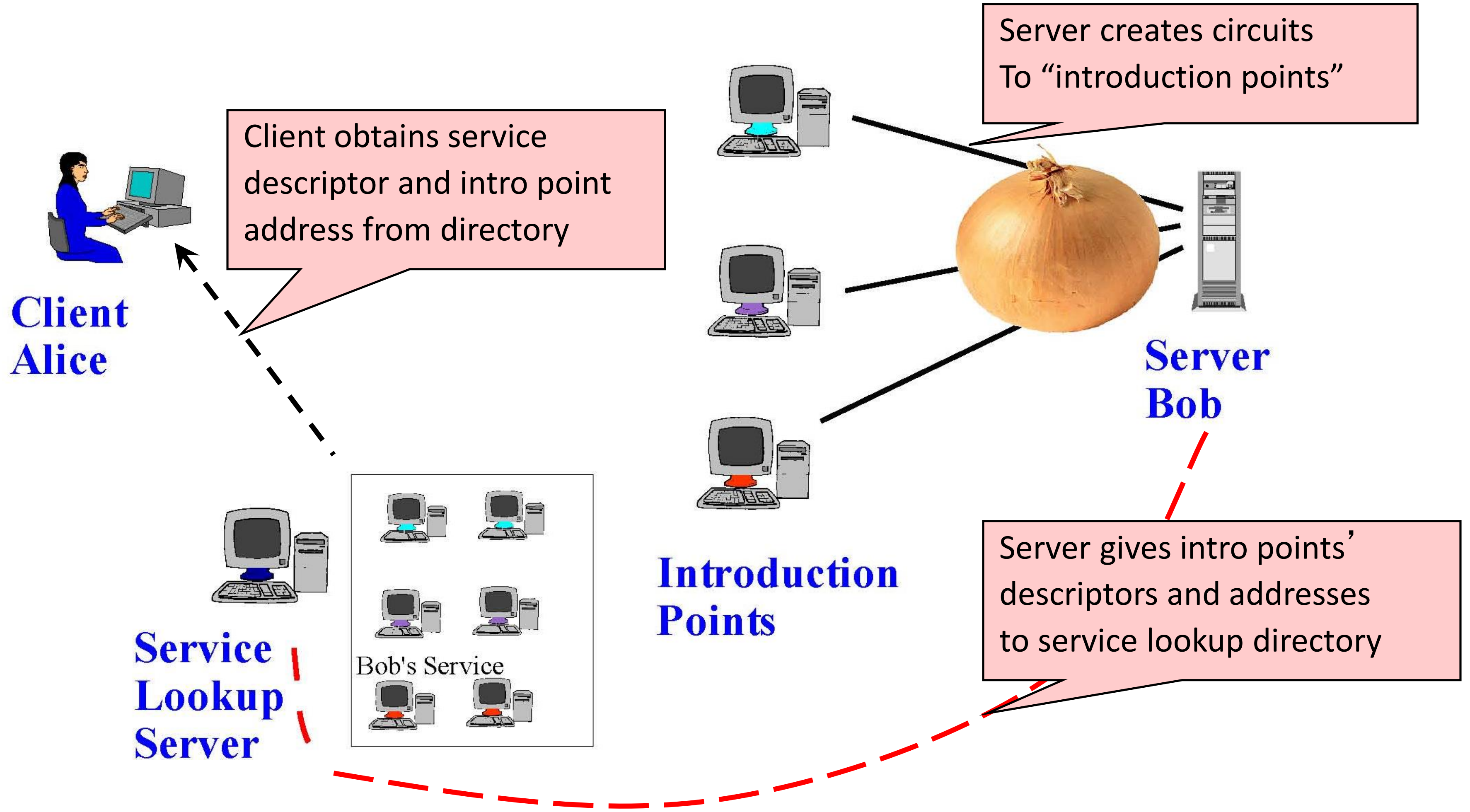
How do you know who to talk to?

- Directory servers
  - Maintain lists of active onion routers, their locations, current public keys, etc.
  - Control how new routers join the network
    - “Sybil attack”: attacker creates a large number of routers
  - Directory servers’ keys ship with Tor code

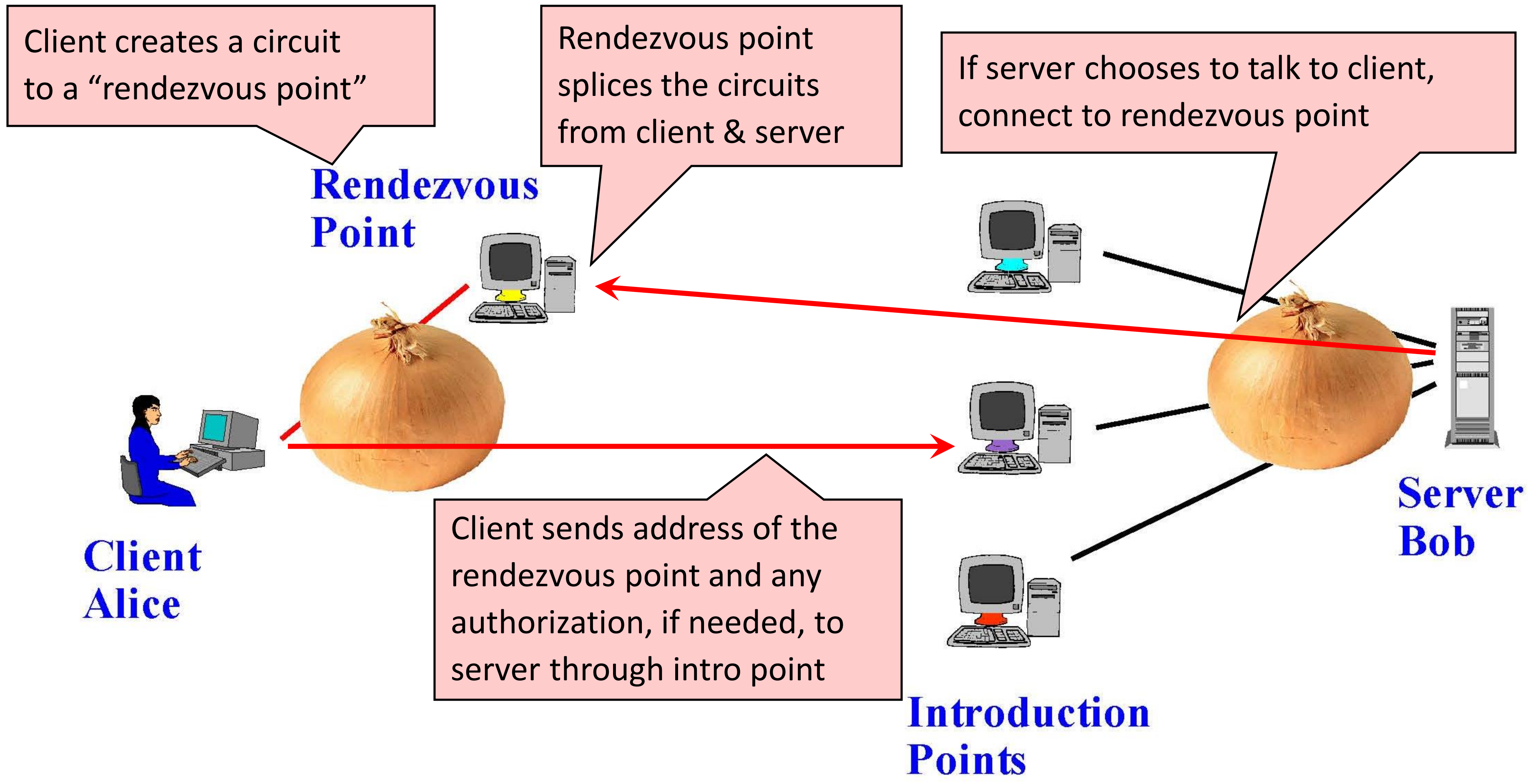
Location Hidden Service

- **Goal:** deploy a server on the Internet that anyone can connect to without knowing where it is or who runs it
- Accessible from anywhere
- Resistant to censorship
- Can survive a full-blown DoS attack
- Resistant to physical attack
    - Can't find the physical server!

Creating a Location Hidden Server


Server creates circuits
To "introduction points"
Client obtains service descriptor and intro point address from directory
Client Alice
Server Bob
Introduction Points
Service Lookup Server
Bob's Service
Server gives intro points' descriptors and addresses to service lookup directory

Using a Location Hidden Server

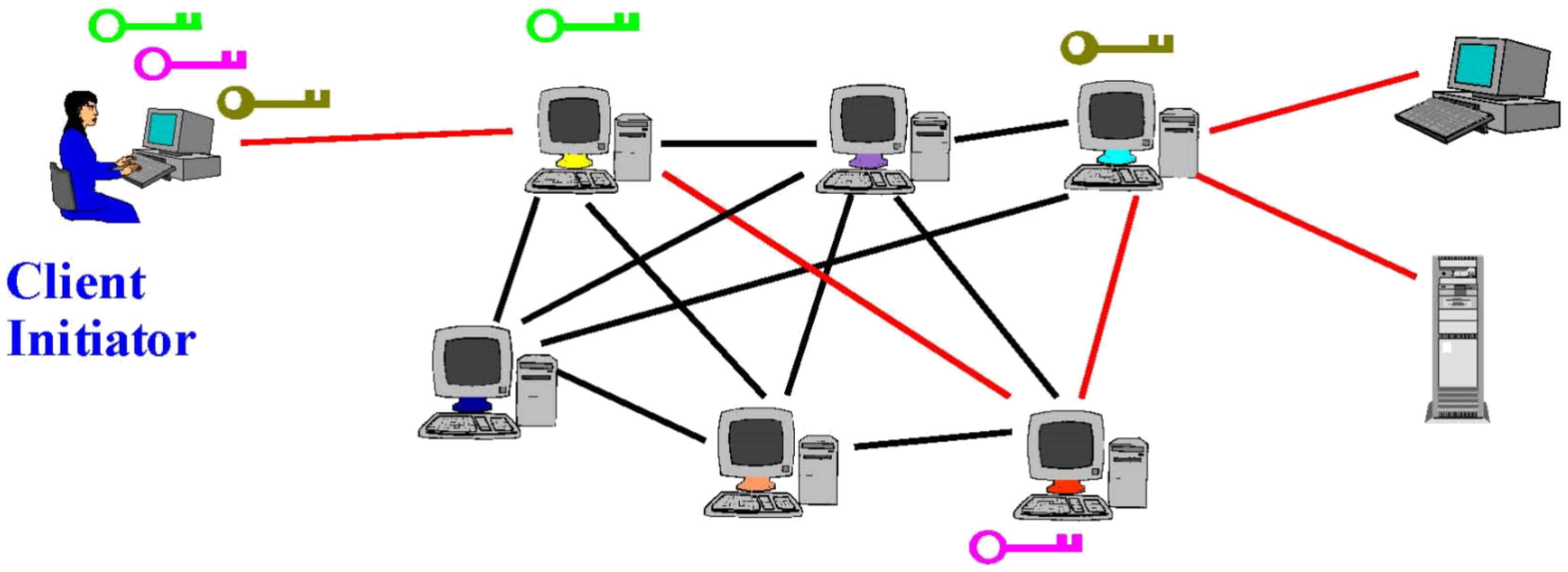

Client creates a circuit to a "rendezvous point"
Rendezvous point splices the circuits from client & server
If server chooses to talk to client, connect to rendezvous point
Rendezvous Point
Client Alice
Server Bob
Client sends address of the rendezvous point and any authorization, if needed, to server through intro point
Introduction Points

Issues and Notes of Caution

- Passive traffic analysis
  - Infer from network traffic who is talking to whom
  - To hide your traffic, must carry other people's traffic!
- Active traffic analysis
  - Inject packets or put a timing signature on packet flow
- Compromise of network nodes
  - Attacker may compromise some routers
    - Powerful adversaries may compromise "too many"
  - It is not obvious which nodes have been compromised
    - Attacker may be passively logging traffic
  - Better not to trust any individual router
    - Assume that some fraction of routers is good, don't know which

Issues and Notes of Caution

- Tor isn't completely effective by itself
    - Tracking cookies, fingerprinting, etc.
    - Exit nodes can see everything!



Issues and Notes of Caution

- The simple act of using Tor could make one a target for additional surveillance
- Hosting an exit node could result in illegal activity coming from your machine
- Tor not designed to protect against adversaries with the capabilities of a state (public statement by designers, at least in the past)

Usability and Security

Aside -- HDCP

Problem: People like copying movies!

- Solution: DRM (Digital Rights Management)
  - DVD players, Streaming service plugins, etc
  - Encrypt video in-transit, decrypt on device

Problem: People like copying movies!

- Solution: DRM (Digital Rights Management)
  - DVD players, Streaming service plugins, etc
  - Encrypt video in-transit, decrypt on device

- Problem: The *analog hole* – You have to display the context eventually

Problem: Analog Hole

- Solution: … The same thing again – DRM
  - HDCP -- High-bandwidth Digital Content Protection
  - Encrypt data on the wire between the computer output and the monitor

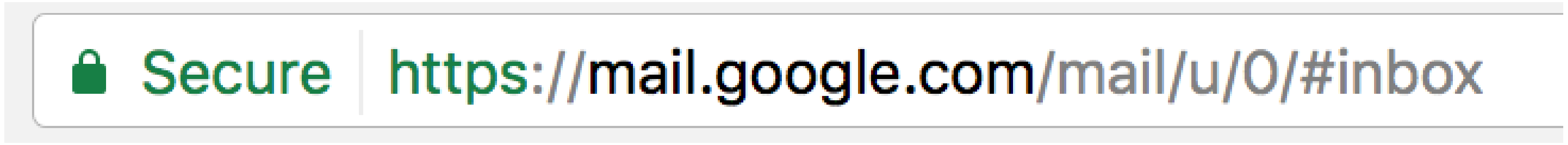
Importance of Usability in Security

- Why is usability important?
  - People are the critical element of any computer system
    - People are the reason computers exist in the first place
  - Even if it is **possible** for a system to protect against an adversary, people may use the system in other, **less secure** ways

Usable Security Roadmap

- 2 case studies
  - HTTPS indicators + SSL warnings – Done in section, will summarize
  - Phishing
- Step back: root causes of usability problems, and how to address

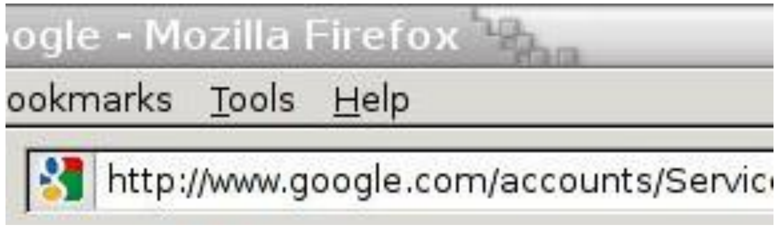
The Lock Icon


Secure https://mail.google.com/mail/u/0/#inbox

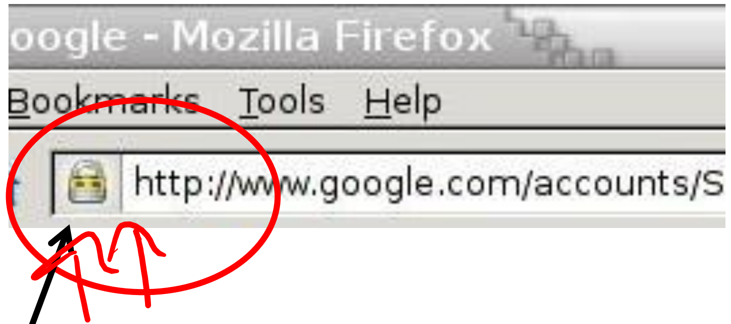
- Goal: identify secure connection
  - SSL/TLS is being used between client and server to protect against active network attacker
- Lock icon should only be shown when the page is secure against network attacker
  - Semantics subtle and not widely understood by users
  - Whose certificate is it??
  - Problem in user interface design

[Moxie Marlinspike]

Will You Notice?



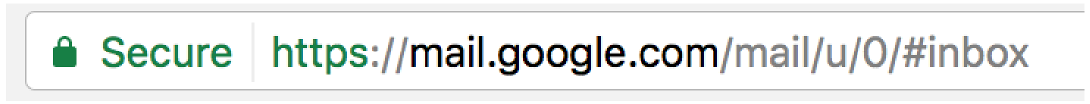
$\Rightarrow$



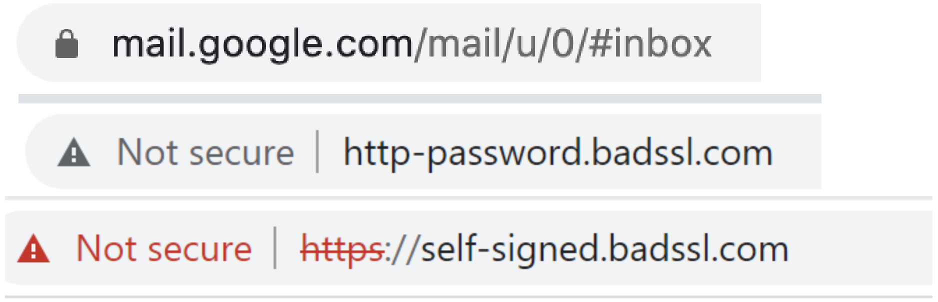
Clever favicon inserted by network attacker

Newer Versions of Chrome

c. 2017


Secure https://mail.google.com/mail/u/0/#inbox

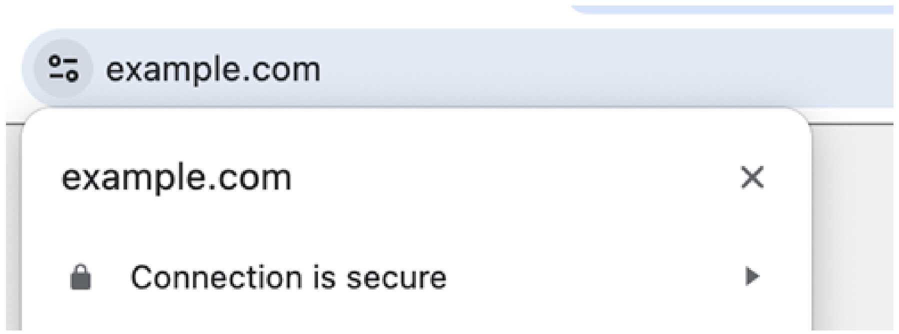
2022


mail.google.com/mail/u/0/#inbox

Not secure | http-password.badssl.com

Not secure | ~~https~~://self-signed.badssl.com

2023/2024

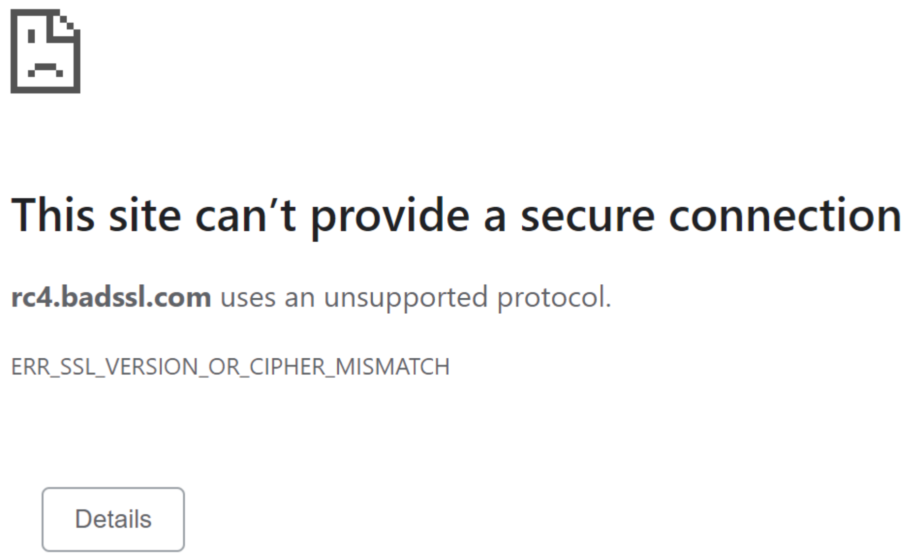

example.com

example.com ✕

Connection is secure

Today’s warnings (2022)

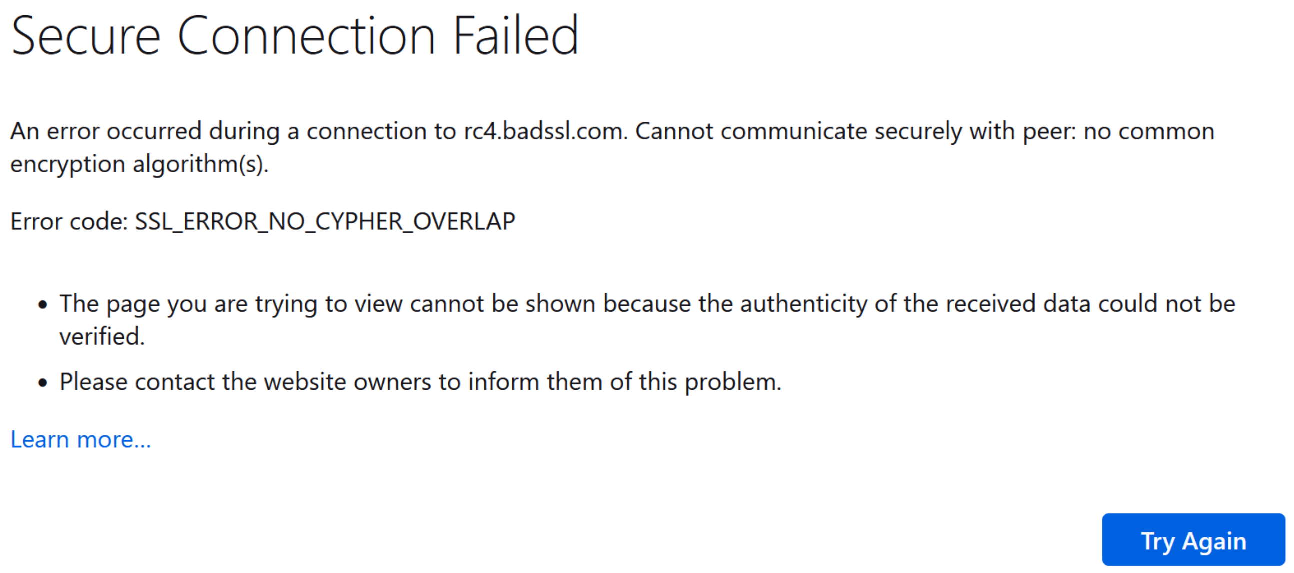
Deprecated encryption schemes



This site can’t provide a secure connection

rc4.badssl.com uses an unsupported protocol.

ERR_SSL_VERSION_OR_CIPHER_MISMATCH



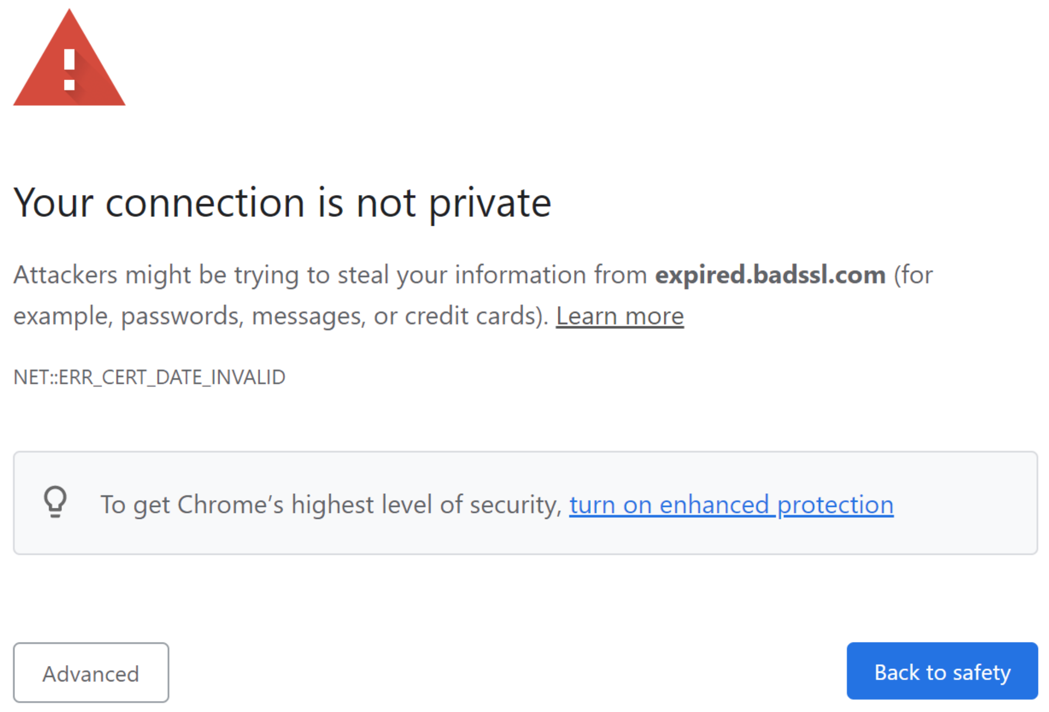
Secure Connection Failed

An error occurred during a connection to rc4.badssl.com. Cannot communicate securely with peer: no common encryption algorithm(s).

Error code: SSL_ERROR_NO_CYPHER_OVERLAP

- The page you are trying to view cannot be shown because the authenticity of the received data could not be verified.
- Please contact the website owners to inform them of this problem.

Expired certificates

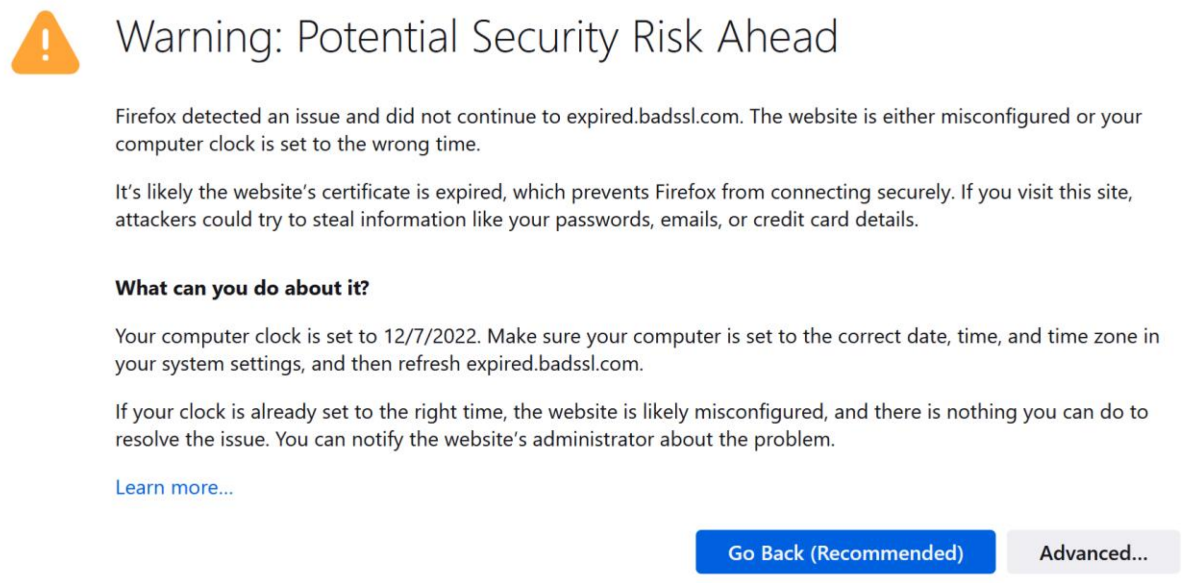


Your connection is not private

Attackers might be trying to steal your information from **expired.badssl.com** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_DATE_INVALID

To get Chrome's highest level of security, turn on enhanced protection



Warning: Potential Security Risk Ahead

Firefox detected an issue and did not continue to expired.badssl.com. The website is either misconfigured or your computer clock is set to the wrong time.

It's likely the website's certificate is expired, which prevents Firefox from connecting securely. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.

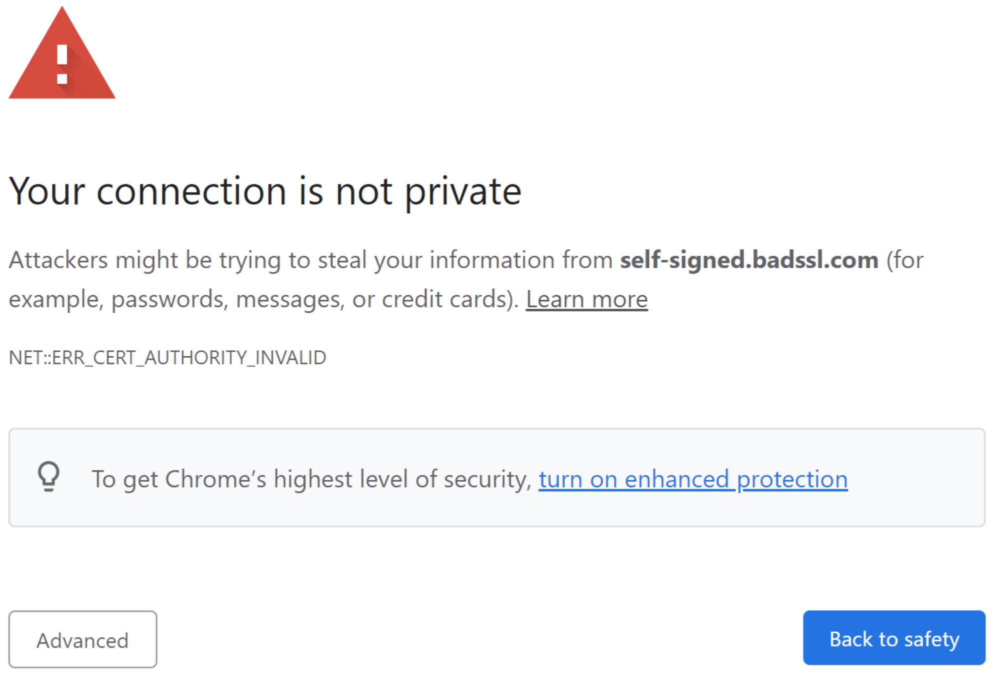
What can you do about it?

Your computer clock is set to 12/7/2022. Make sure your computer is set to the correct date, time, and time zone in your system settings, and then refresh expired.badssl.com.

If your clock is already set to the right time, the website is likely misconfigured, and there is nothing you can do to resolve the issue. You can notify the website's administrator about the problem.

Learn more...

Self-signed certificates

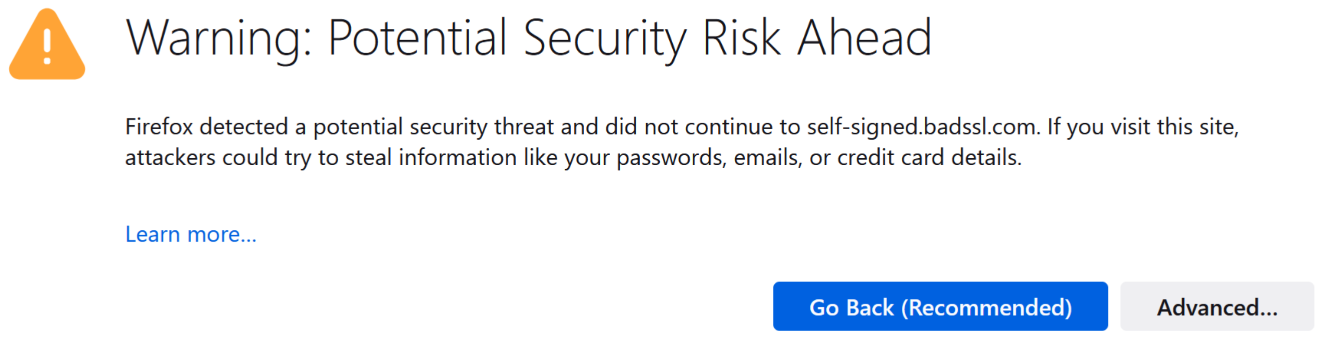


Your connection is not private

Attackers might be trying to steal your information from **self-signed.badssl.com** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_AUTHORITY_INVALID

To get Chrome's highest level of security, turn on enhanced protection

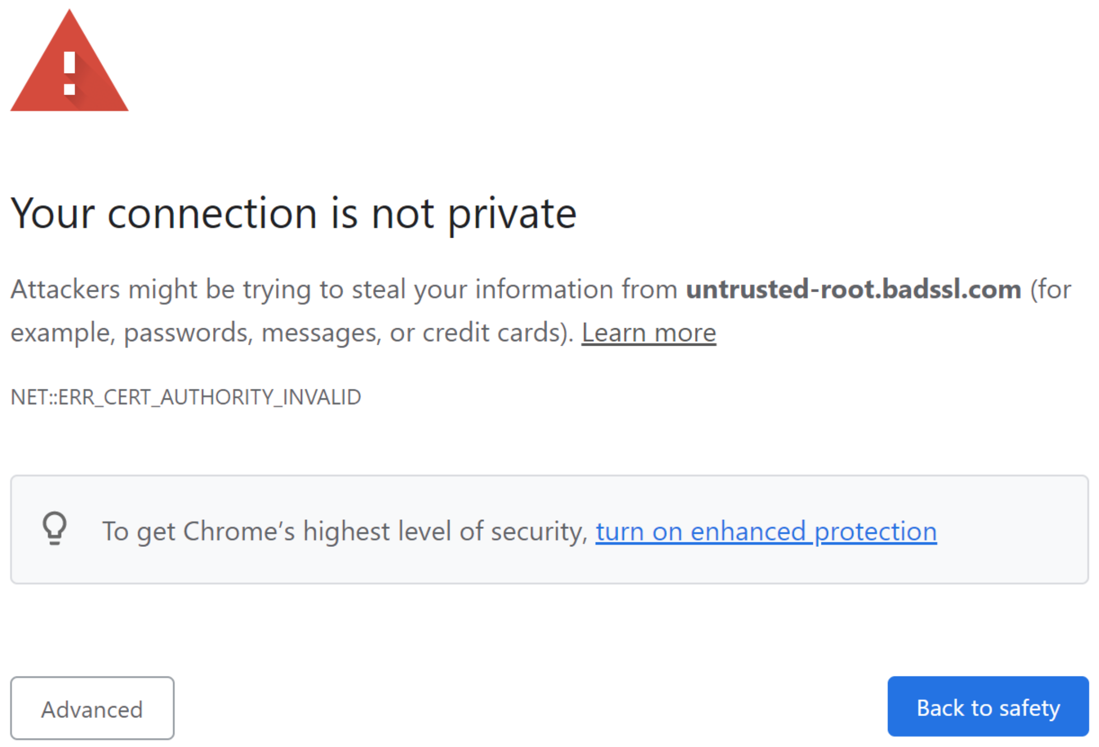


Warning: Potential Security Risk Ahead

Firefox detected a potential security threat and did not continue to self-signed.badssl.com. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.

Learn more...

Untrusted Root certificate

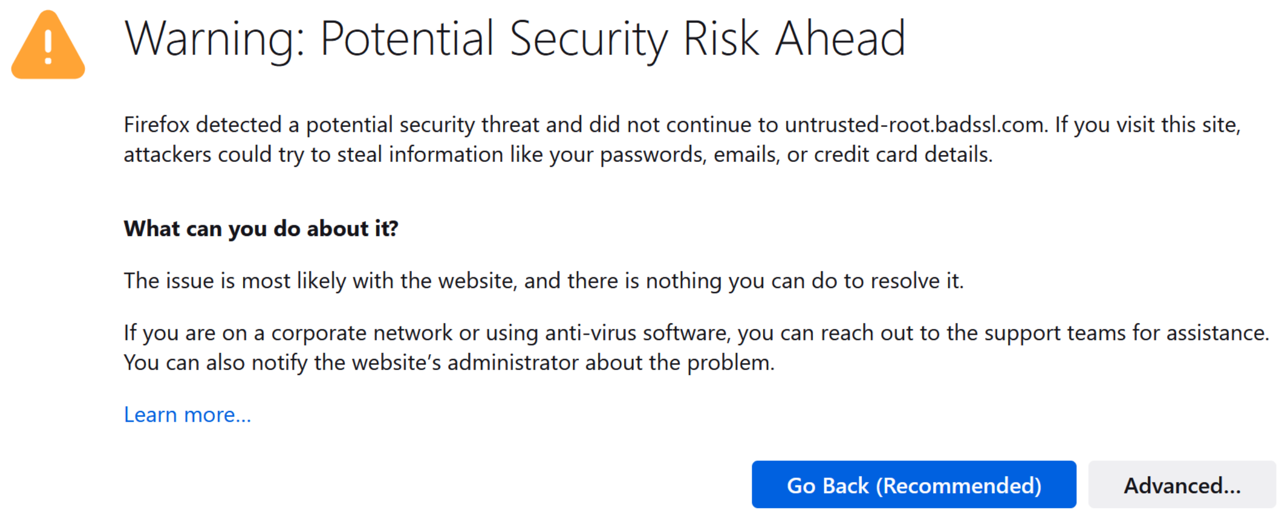


Your connection is not private

Attackers might be trying to steal your information from **untrusted-root.badssl.com** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_AUTHORITY_INVALID

To get Chrome's highest level of security, turn on enhanced protection



Warning: Potential Security Risk Ahead

Firefox detected a potential security threat and did not continue to untrusted-root.badssl.com. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.

What can you do about it?

The issue is most likely with the website, and there is nothing you can do to resolve it.

If you are on a corporate network or using anti-virus software, you can reach out to the support teams for assistance. You can also notify the website's administrator about the problem.

Learn more...

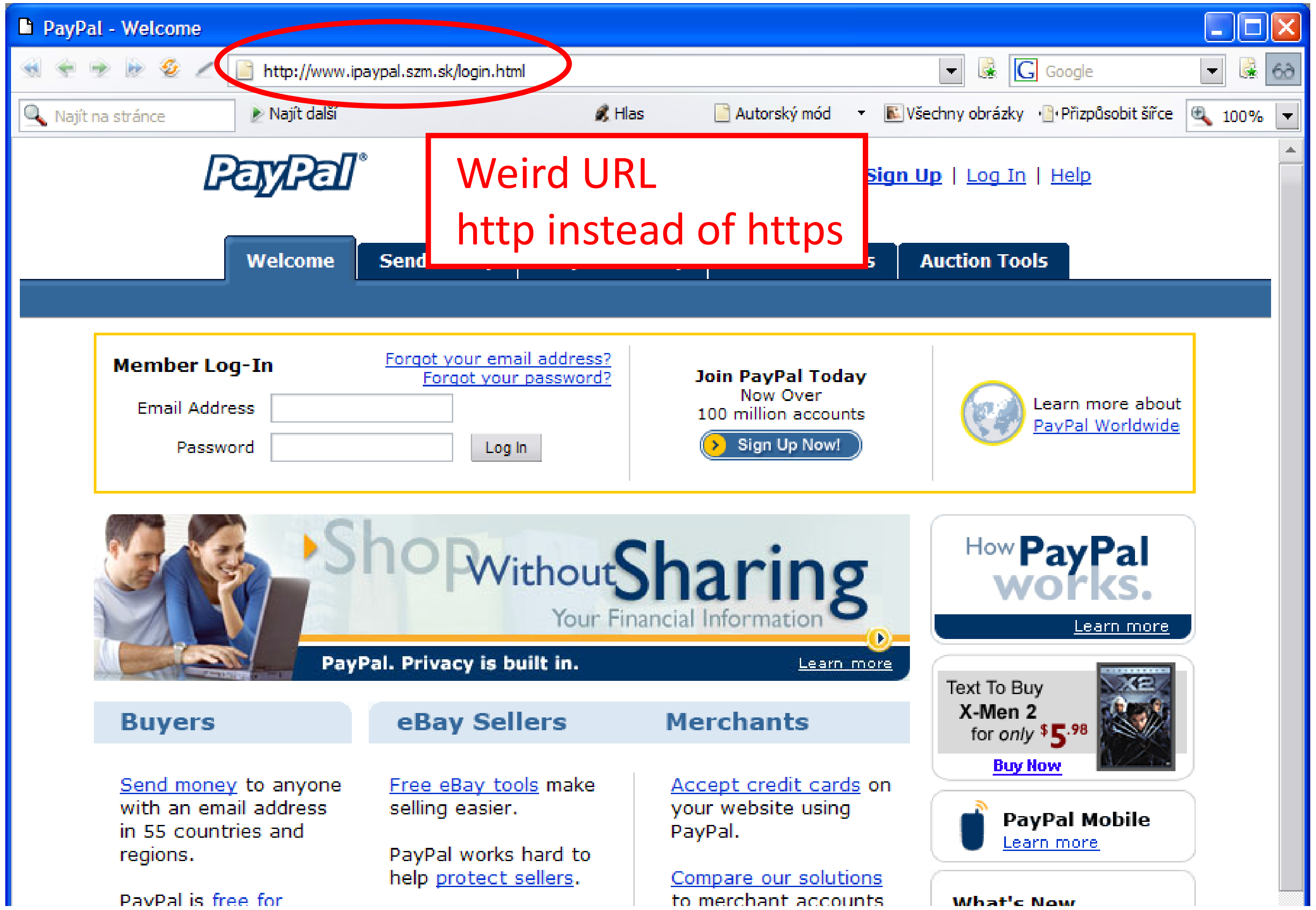
Does anything stand out?

- Gradescope:
- Q1: What are some things that make warnings hard to be effective?
- Q2: Why would Firefox and Chrome choose different warning designs?

Case Study #2: Phishing

- **Design question:** How do you help users avoid falling for phishing sites?

A Typical Phishing Page



Send money to anyone with an email address in 55 countries and regions.

PayPal is free for

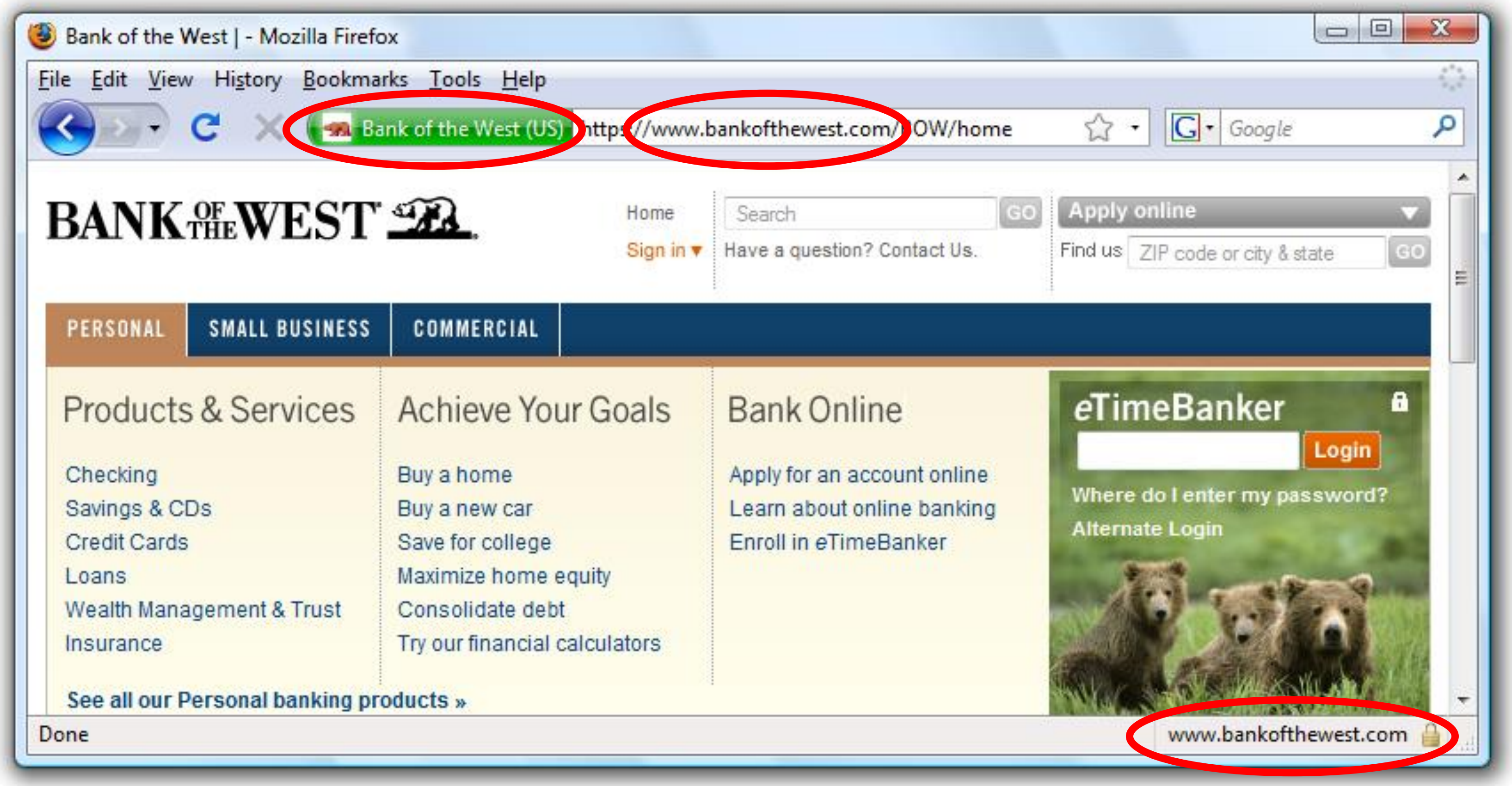
Free eBay tools make selling easier.

PayPal works hard to help protect sellers.

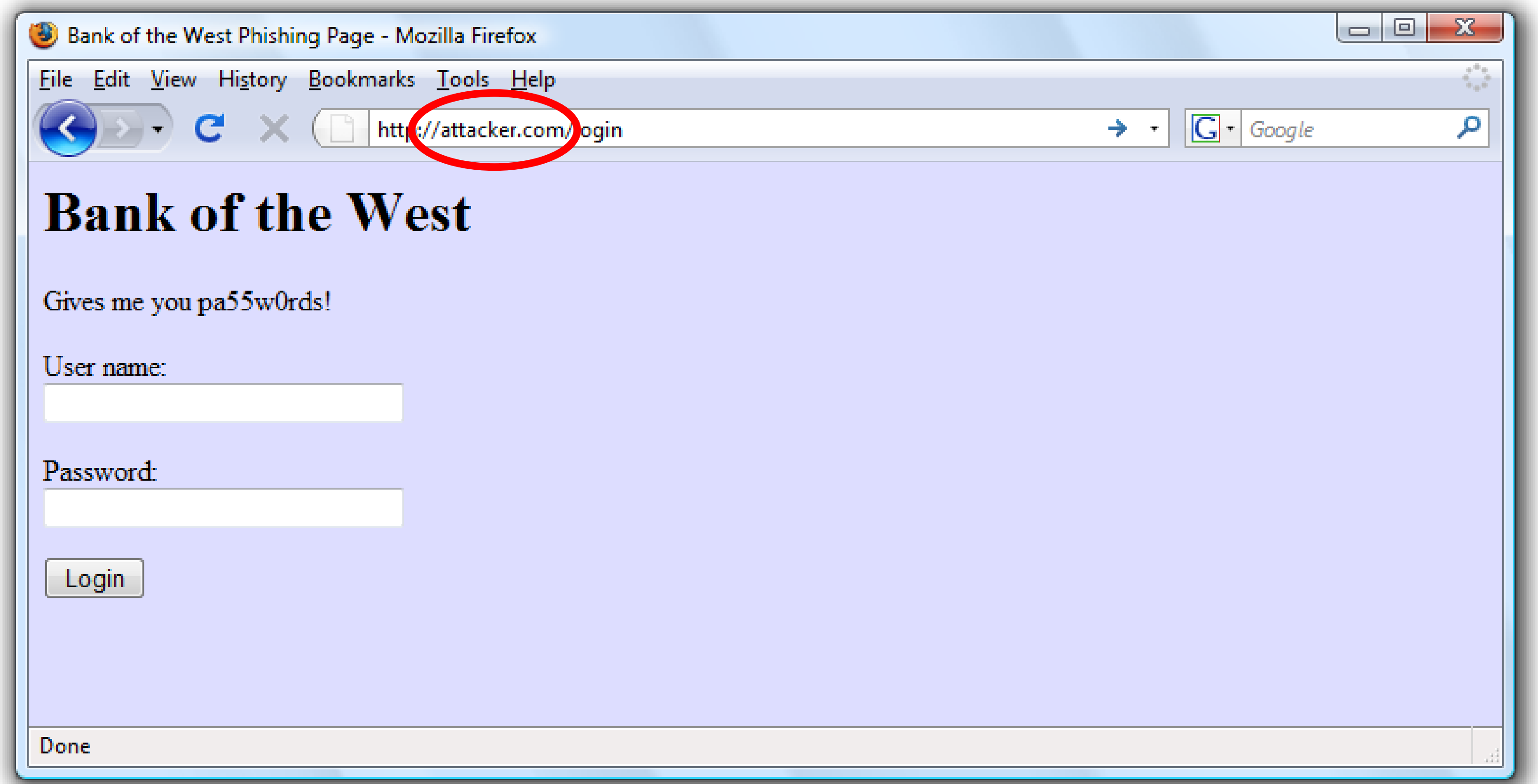
Accept credit cards on your website using PayPal.

Compare our solutions to merchant accounts

Safe to Type Your Password?


Bank of the West | - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Bank of the West (US) https://www.bankofthewest.com/BOW/home
Google
BANK OF THE WEST
Home
Sign in
Search GO
Have a question? Contact Us.
Apply online
Find us ZIP code or city & state GO
PERSONAL
SMALL BUSINESS
COMMERCIAL
Products & Services
Checking
Savings & CDs
Credit Cards
Loans
Wealth Management & Trust
Insurance
Achieve Your Goals
Buy a home
Buy a new car
Save for college
Maximize home equity
Consolidate debt
Try our financial calculators
Bank Online
Apply for an account online
Learn about online banking
Enroll in eTimeBanker
eTimeBanker
Login
Where do I enter my password?
Alternate Login
See all our Personal banking products »
Done
www.bankofthewest.com

Safe to Type Your Password?


Bank of the West Phishing Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://attacker.com/login
Google
Login
Done

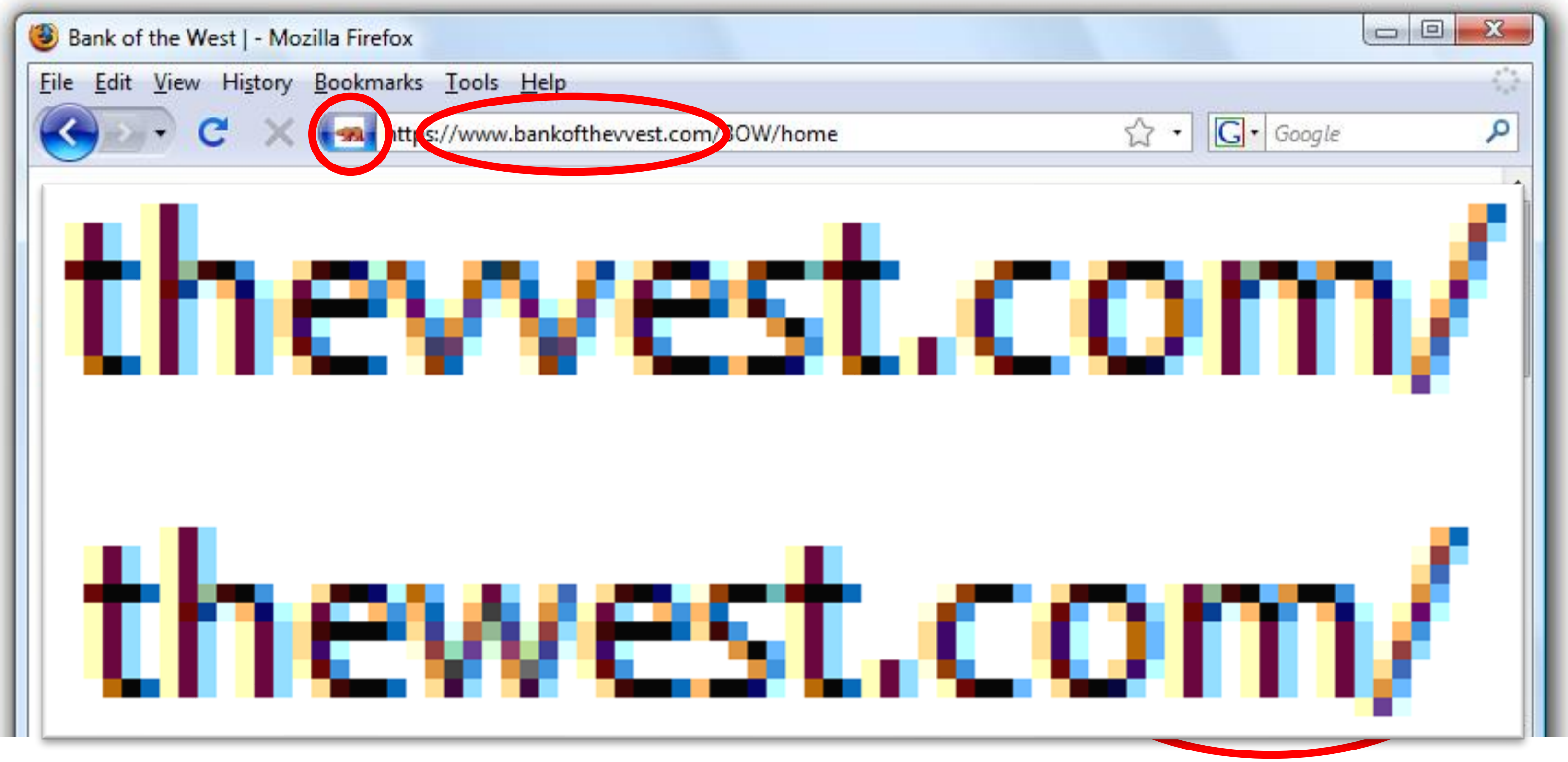
Bank of the West

Gives me you pa55w0rds!

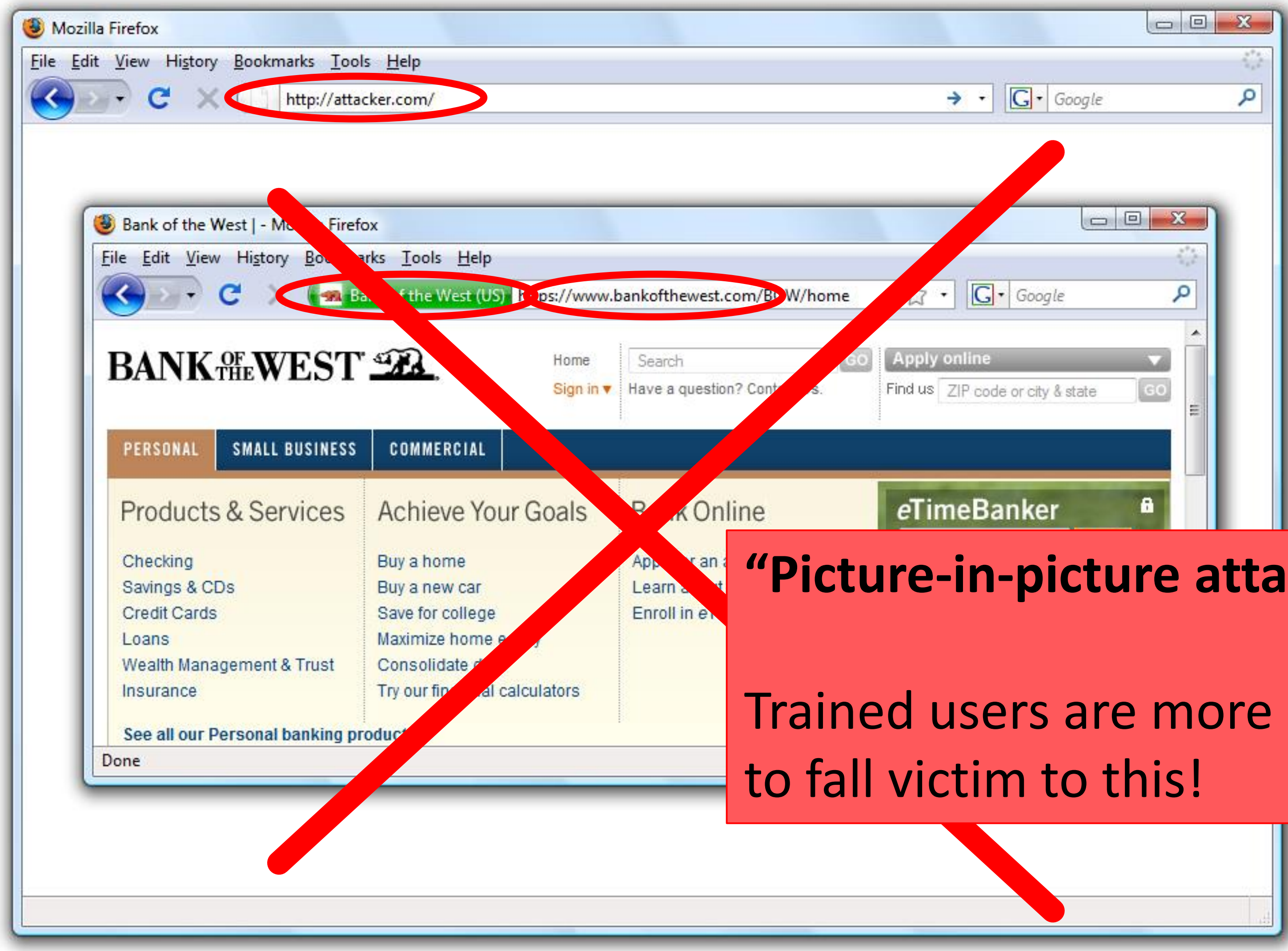
User name:

Password:

Safe to Type Your Password?


Bank of the West | - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://www.bankofthevvest.com/BOW/home
Google
thevvest.com/
thewest.com/

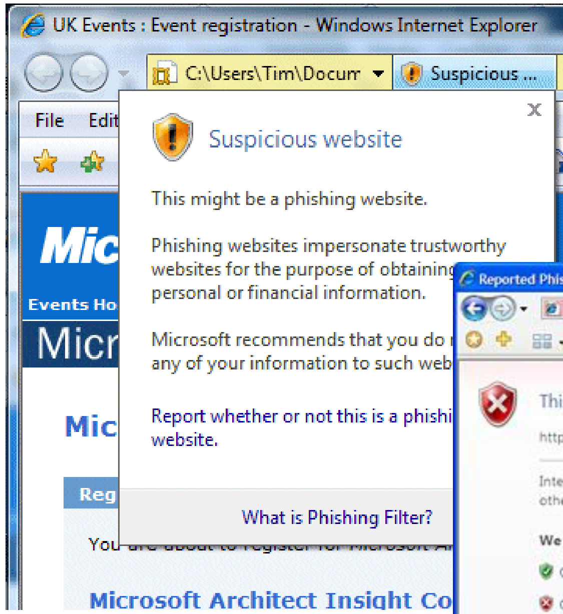
Safe to Type Your Password?



“Picture-in-picture attacks”

Trained users are more likely to fall victim to this!

Phishing Warnings (2008)



UK Events : Event registration - Windows Internet Explorer

C:\Users\Tim\Docum

Suspicious ...

Suspicious website

This might be a phishing website.

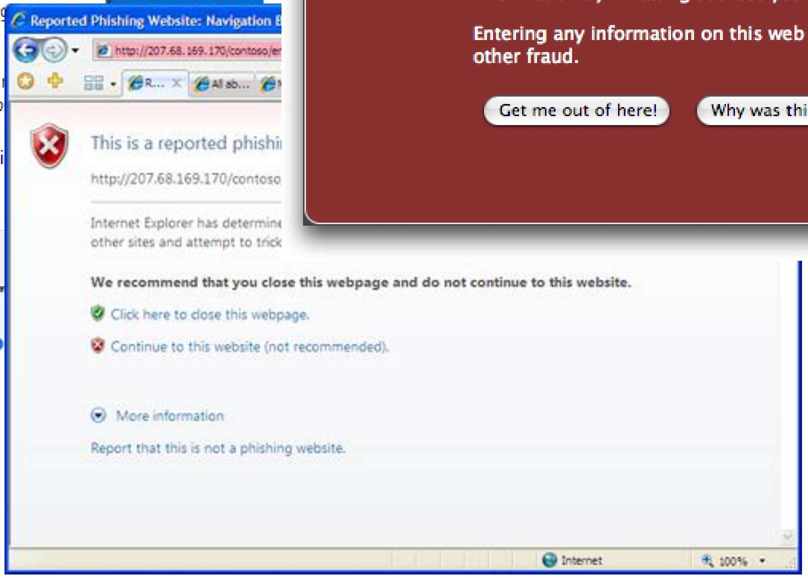
Phishing websites impersonate trustworthy websites for the purpose of obtaining personal or financial information.

Microsoft recommends that you do any of your information to such web

Report whether or not this is a phishi website.

What is Phishing Filter?

Passive (IE)



Reported Phishing Website: Navigation

http://207.68.169.170/contoso/

This is a reported phishi

http://207.68.169.170/contoso

Internet Explorer has determine other sites and attempt to trick

We recommend that you close this webpage and do not continue to this website.

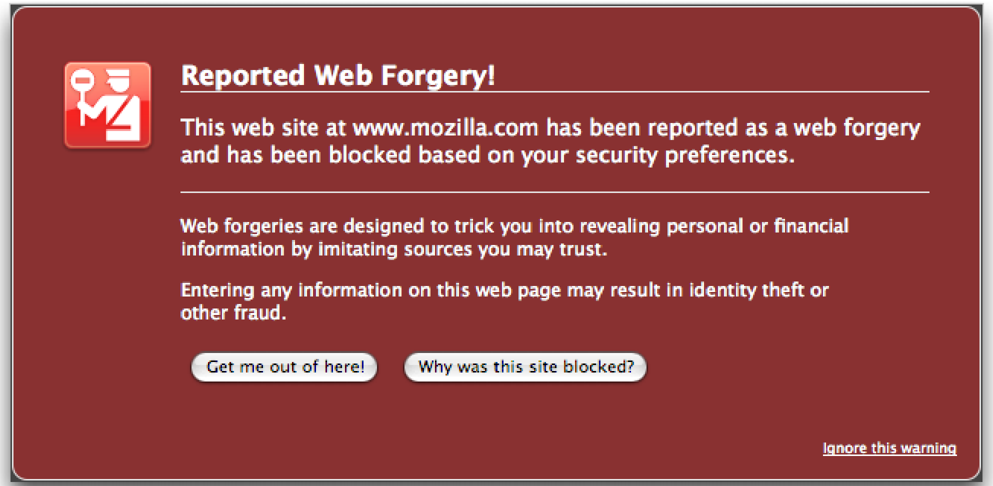
Click here to close this webpage.

Continue to this website (not recommended).

More information

Report that this is not a phishing website.

Active (IE)



Reported Web Forgery!

This web site at www.mozilla.com has been reported as a web forgery and has been blocked based on your security preferences.

Web forgeries are designed to trick you into revealing personal or financial information by imitating sources you may trust.

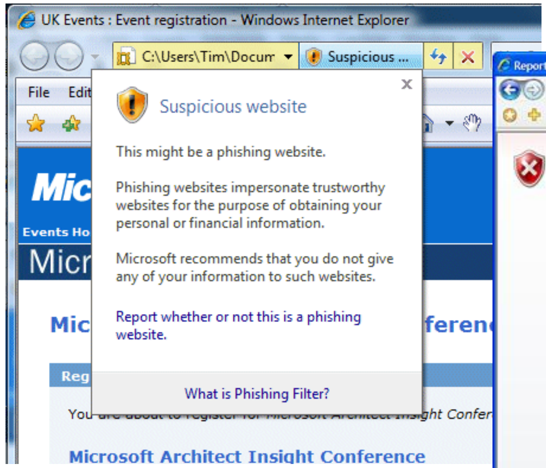
Entering any information on this web page may result in identity theft or other fraud.

Active (Firefox)

[Egelman et al.]

Active vs. Passive Warnings

- Active warnings significantly more effective
  - Passive (IE): 100% clicked, 90% phished
  - Active (IE): 95% clicked, 45% phished
  - Active (Firefox): 100% clicked, 0% phished



UK Events : Event registration - Windows Internet Explorer

Suspicious website

This might be a phishing website.

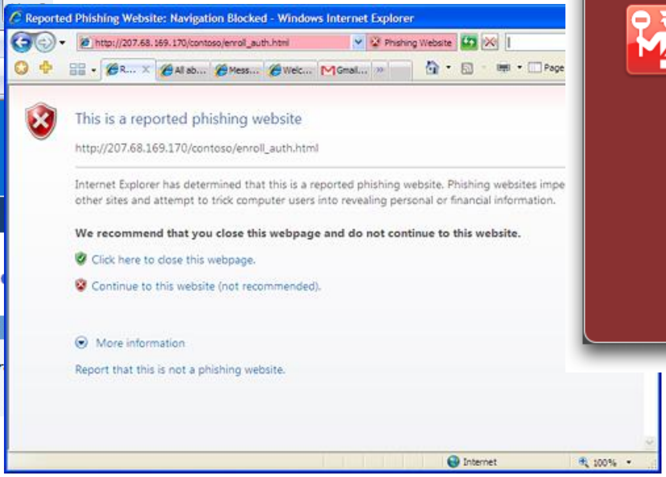
Phishing websites impersonate trustworthy websites for the purpose of obtaining your personal or financial information.

Microsoft recommends that you do not give any of your information to such websites.

Report whether or not this is a phishing website.

What is Phishing Filter?

Passive (IE)



Reported Phishing Website: Navigation Blocked - Windows Internet Explorer

This is a reported phishing website

http://207.68.169.170/contoso/enroll_auth.html

Internet Explorer has determined that this is a reported phishing website. Phishing websites impe
other sites and attempt to trick computer users into revealing personal or financial information.

We recommend that you close this webpage and do not continue to this website.

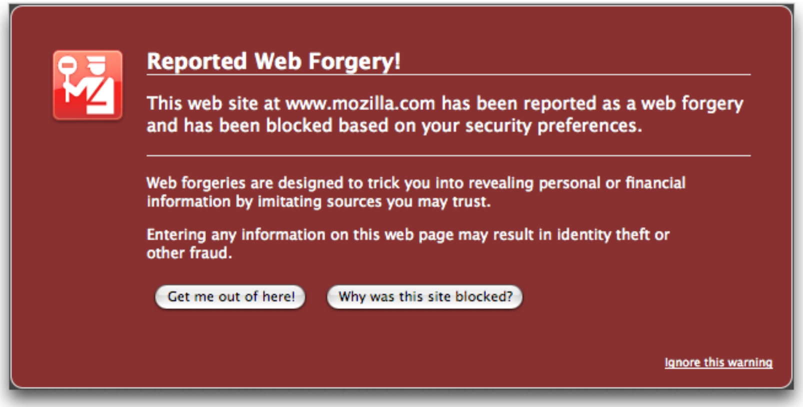
Click here to close this webpage.

Continue to this website (not recommended).

More information

Report that this is not a phishing website.

Active (IE)



Reported Web Forgery!

This web site at www.mozilla.com has been reported as a web forgery and has been blocked based on your security preferences.

Web forgeries are designed to trick you into revealing personal or financial information by imitating sources you may trust.

Entering any information on this web page may result in identity theft or other fraud.

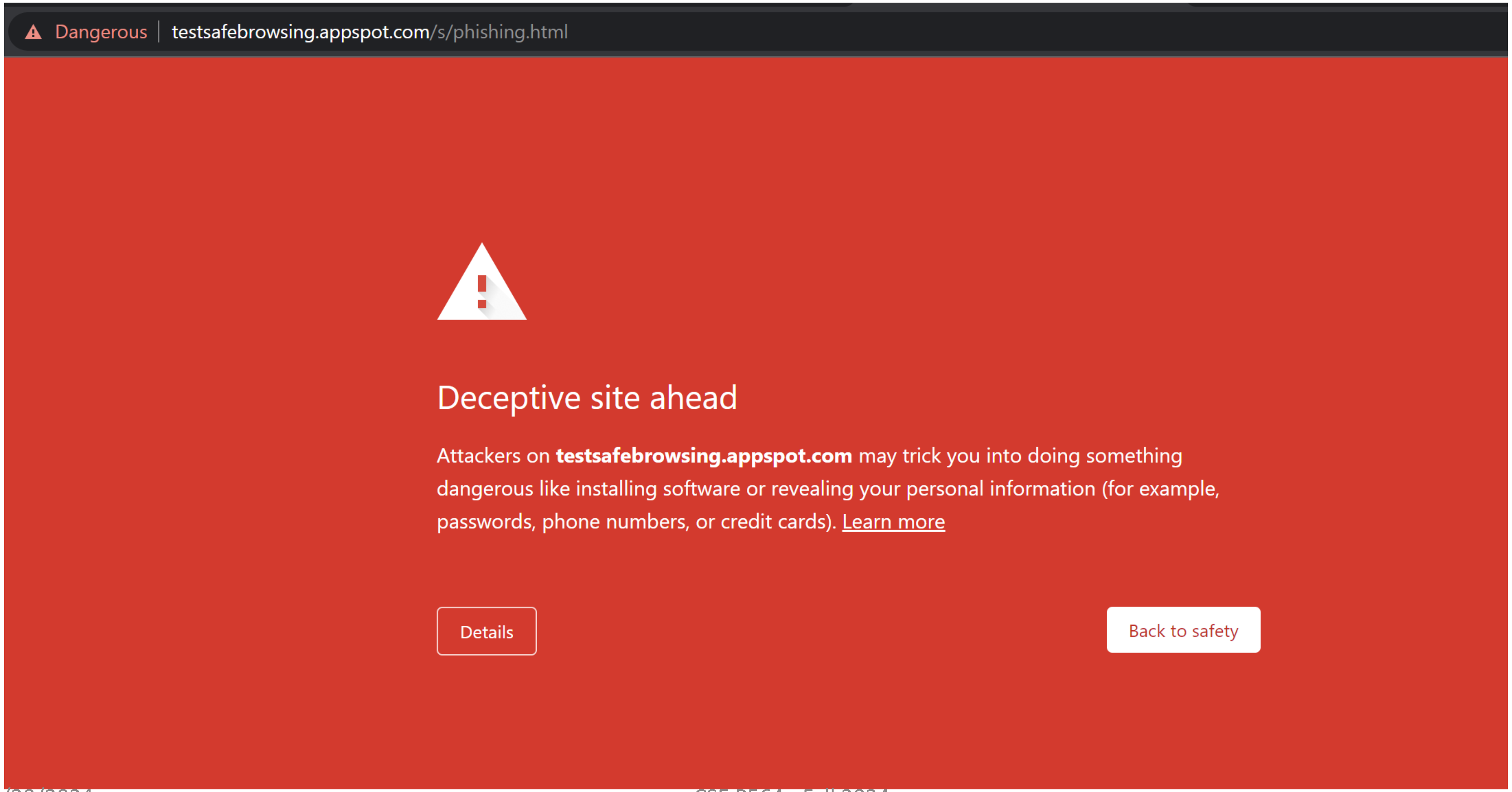
Active (Firefox)

Modern anti-phishing

- Largely driven by Google Safe Browsing
  - Browser sends 32-bit prefix of hash(url)
  - API says: good or bad

- (Also Microsoft SafeScreen)

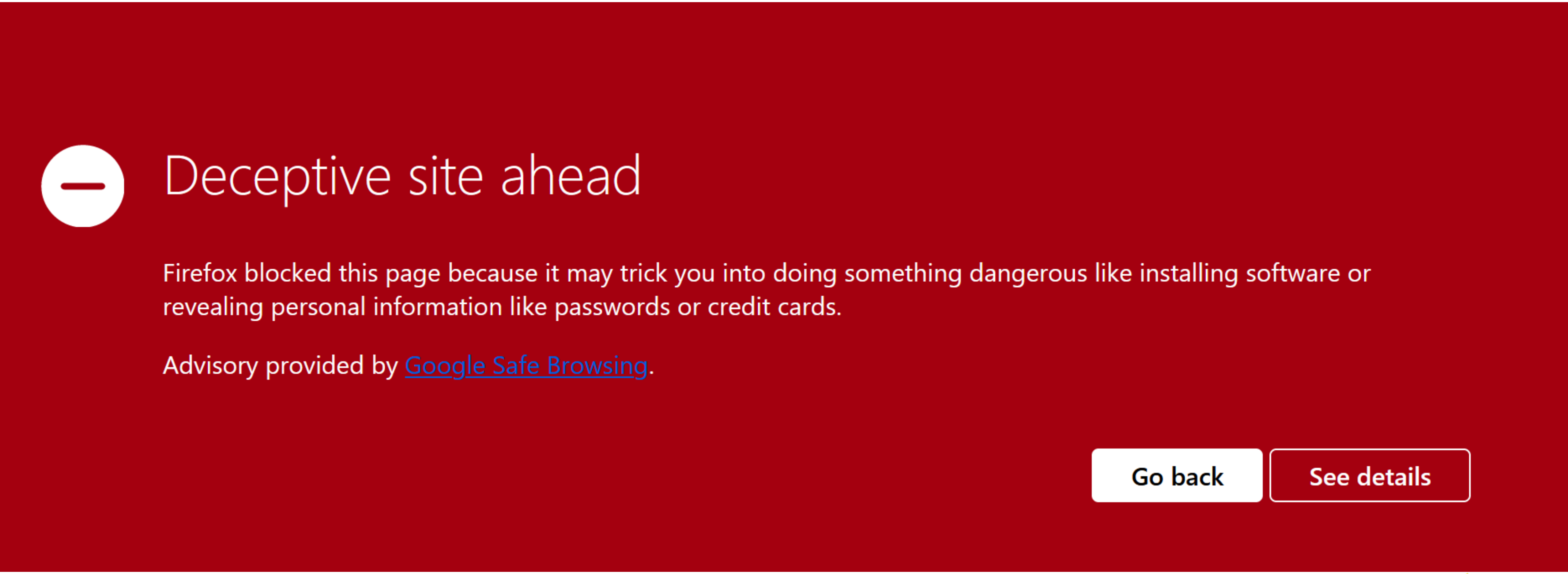
Modern warnings



Dangerous | testsafebrowsing.appspot.com/s/phishing.html

Deceptive site ahead

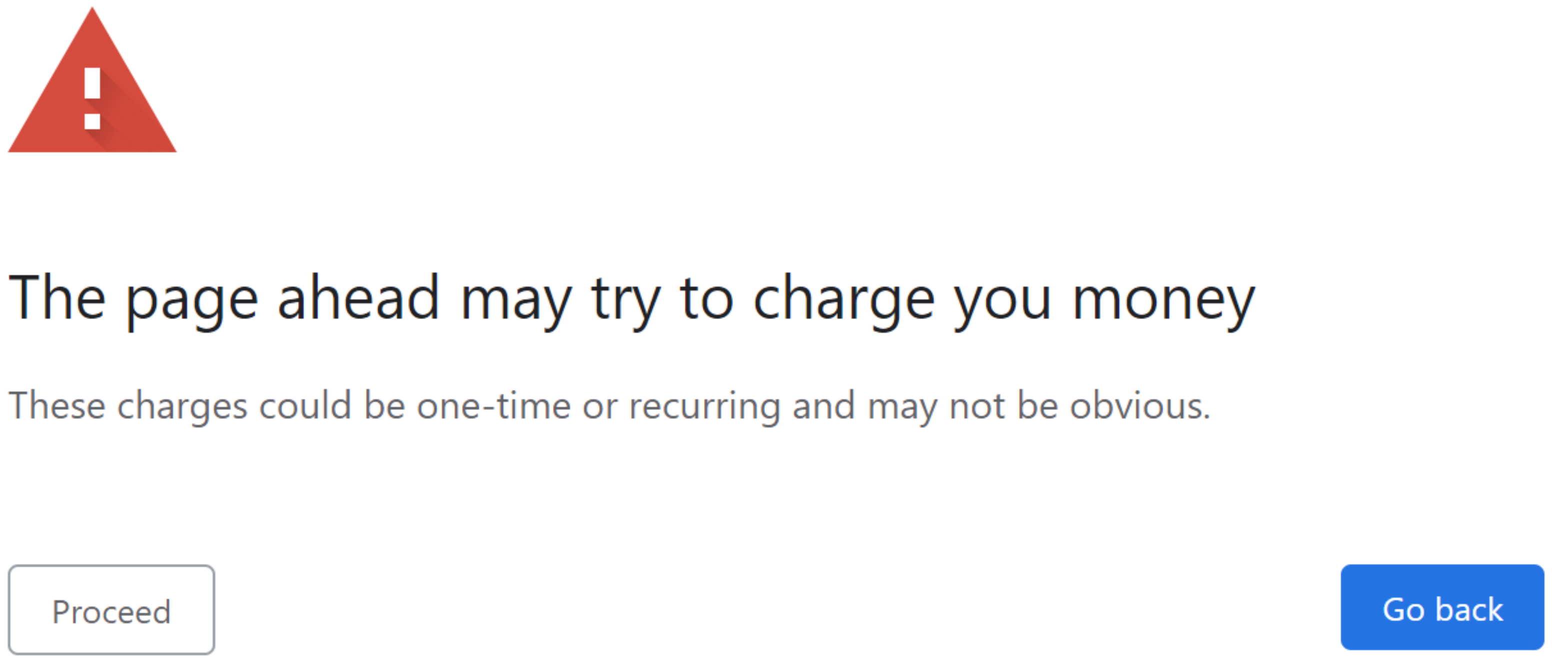
Attackers on **testsafebrowsing.appspot.com** may trick you into doing something dangerous like installing software or revealing your personal information (for example, passwords, phone numbers, or credit cards). Learn more



Deceptive site ahead

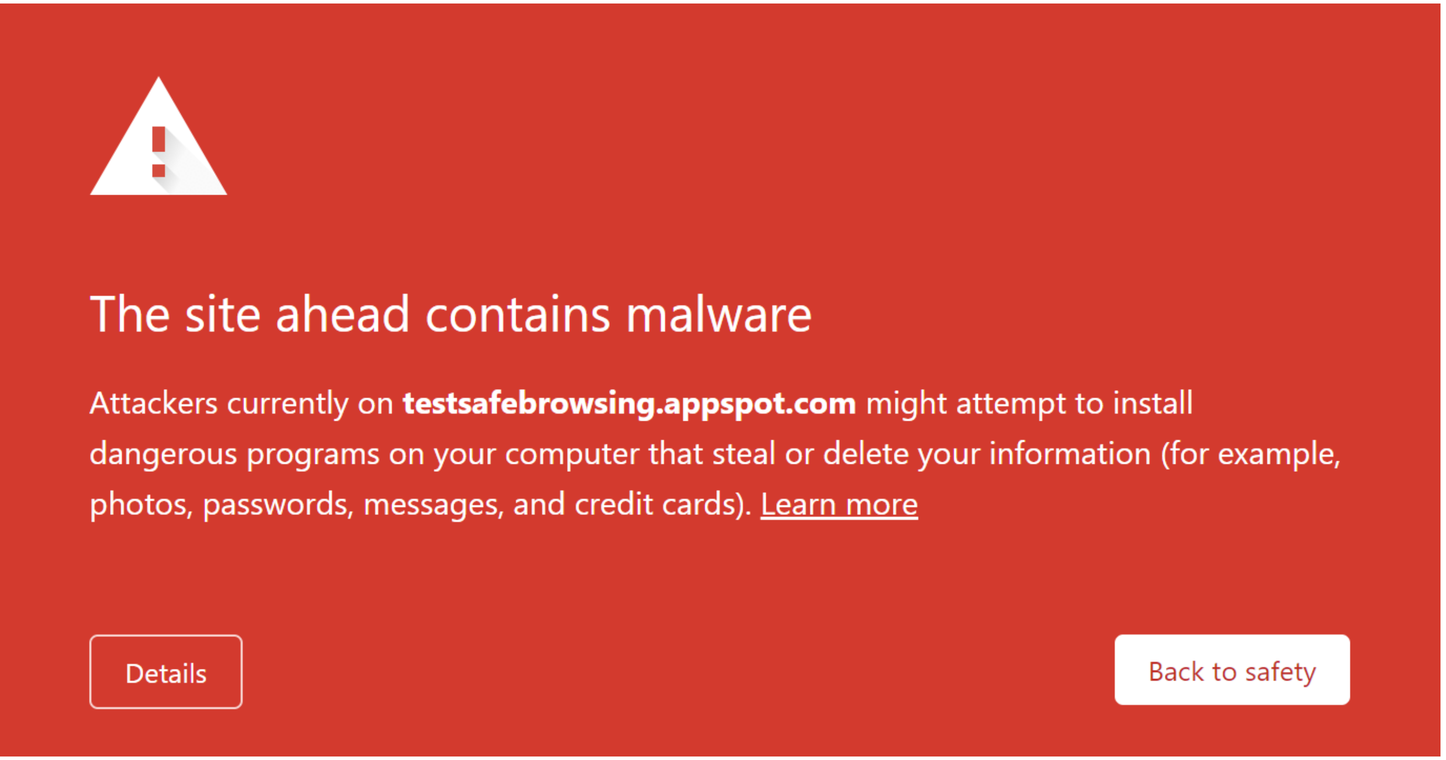
Firefox blocked this page because it may trick you into doing something dangerous like installing software or revealing personal information like passwords or credit cards.

Advisory provided by Google Safe Browsing.

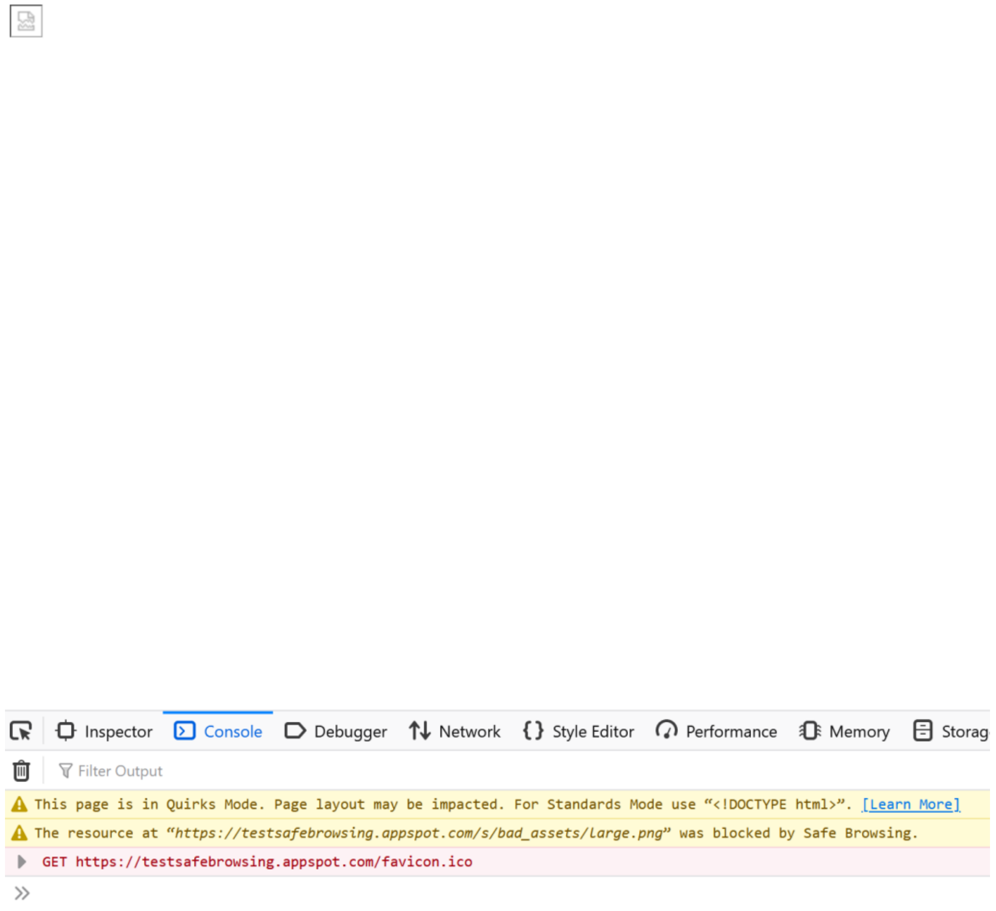


The page ahead may try to charge you money

These charges could be one-time or recurring and may not be obvious.


The site ahead contains malware

Attackers currently on **testsafebrowsing.appspot.com** might attempt to install dangerous programs on your computer that steal or delete your information (for example, photos, passwords, messages, and credit cards). Learn more



```
This page is in Quirks Mode. Page layout may be impacted. For Standards Mode use "<!DOCTYPE html>". [Learn More]
The resource at "https://testsafebrowsing.appspot.com/s/bad_assets/large.png" was blocked by Safe Browsing.
GET https://testsafebrowsing.appspot.com/favicon.ico
```

Which warning is ‘better’?

- For user security?
- For user agency?
- For user understanding?
- For… what?