Security for Web Applications

Web Vulnerabilities: HTML & DOM

References

- *Securing Frame Communication in Browsers*. Adam Barth, Collin Jackson, and John C. Mitchell

  http://seclab.stanford.edu/websec/frames/post-message.pdf

- Same Origin Policy

  http://code.google.com/p/browsersec/wiki/Part2#Same-origin_policy

- Cross Site Request Forgery

  http://users.ece.cmu.edu/~dbrumley/courses/18732-f09/2009-10-14-csrf-cmu-ece-18732.pdf

- XSS Tutorial

  http://hackertarget.com/xss-tutorial/

Web Security: Overview

Web Applications

- **Software as a (Web-based) service:**
  - Online banking, shopping, government, etc.
  - Cloud computing.
- **Languages for web applications:**
  - A mixture of HTML, PHP, Java, Perl, Python, C, ASP.
- **Security is rarely the main concern:**
  - Poorly written scripts with inadequate input validation
  - Sensitive data stored in world-readable files

Typical Web Application Design

- Takes ***input*** from Web users (via Web server).
- Interacts with back-end databases and third parties.
- Prepares and ***outputs*** results for users (via Web server)
  - Dynamically generated HTML pages
  - Content from many different sources/registered users:
    - Blogs, social networks, photo-sharing websites…

Browser: Execution

- Interface between user and server.

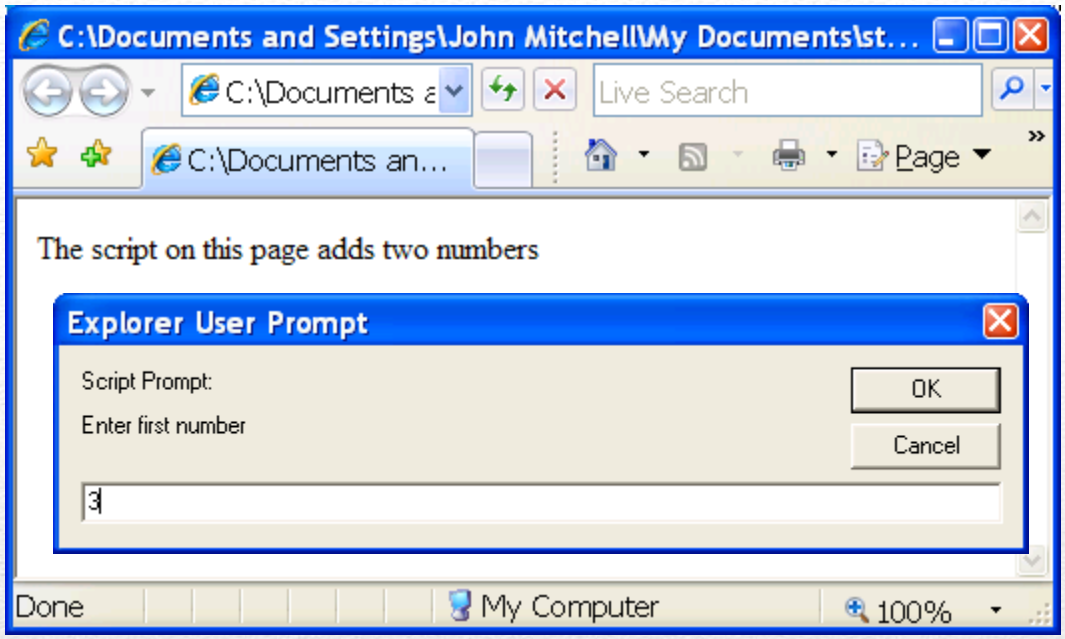
1. Loads content (output from web server).

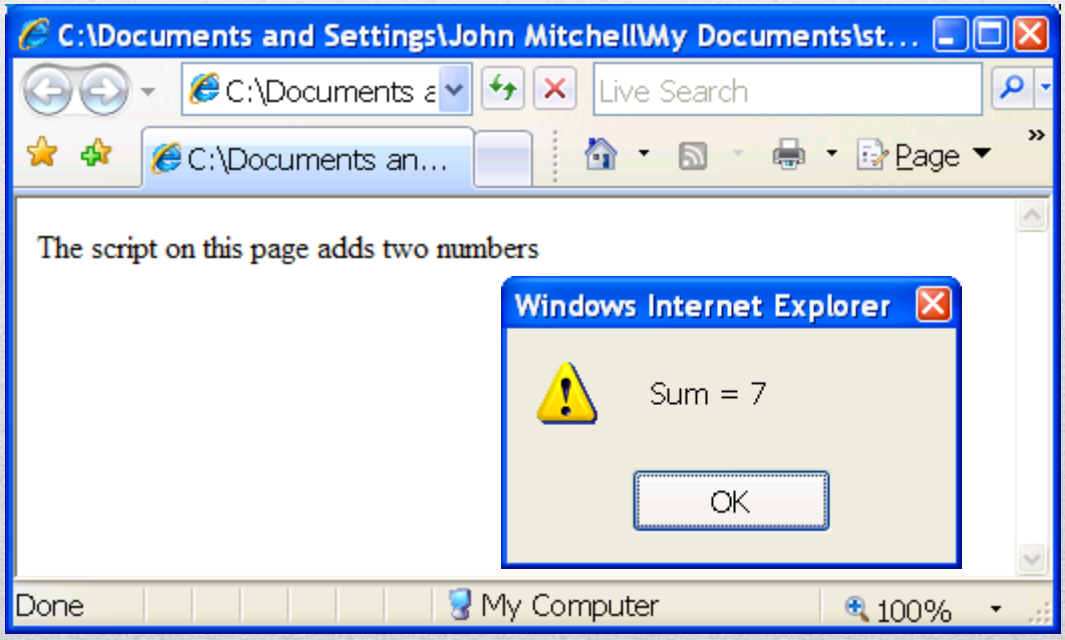
2. Renders:
   - Processes **HTML** and **Scripts** to display the page.
   - May involve images, subframes, etc.

3. Responds to **events:**
   - User actions:
     - OnClick, OnMouseover, OnKeyPress, OnMouseDown
   - Rendering:
     - OnLoad, OnUnLoad
   - Timing:
     - setTimeout, clearTimeout

HTML/Script Example

```
<html>
    …
<p> The script on this page adds two numbers </p>
<script>
                    var num1, num2, sum
                    num1 = prompt("Enter first number")
                    num2 = prompt("Enter second number")
                    sum = parseInt(num1) + parseInt(num2)
                    alert("Sum = " + sum)
</script>
                    …
</html>
```





Try on: http://www.w3schools.com/htmldom/tryit.asp?filename=try_methods

Event-Driven Script

```
<script type="text/javascript">
  function whichButton(event) {
        if (event.which==1) alert("You clicked the left
mouse button!")
        else  alert("You clicked the right mouse button!")
 }
</script>
…
<body OnMouseDown="whichButton(event)">
…
</body>
```

JavaScript into HTML pages

- Embedded in HTML page as SCRIPT:
  - directly inside: <script> alert("Hello World!") </script>
  - Linked file as *src* attribute:

    <script type="text/JavaScript" src="functions.js"></script>

- Event handler attribute:

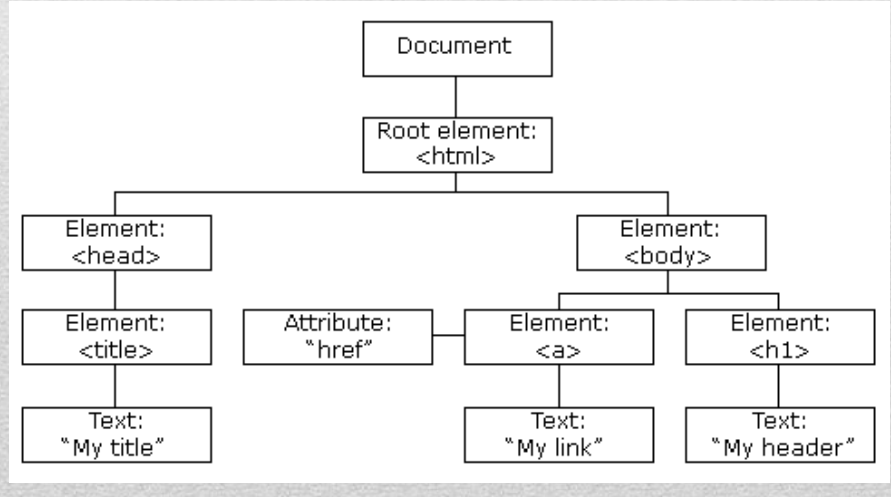
  <a href=http://www.yahoo.com OnMouseOver=alert('hi');>

- Pseudo-URL referenced by a link

  Click me

HTML Structure: DOM

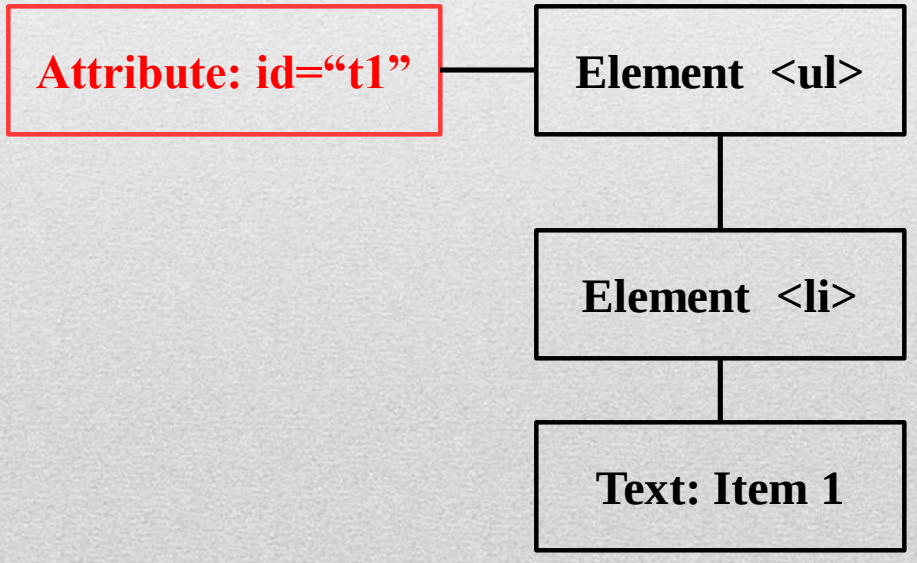
- **Document Object Model (DOM):**
  - Document nodes: elements, attributes, texts/comments
  - Elements defined as objects (accessed with JavaScript, etc.)
    - Via ***method***: action like add, modify an object
      - getElementById(), createElement(), getAttribute()
    - Via ***property***: states like name, content
      - x.firstChild.nodeValue, .innerHTML



DOM: Example

Sample HTML

```
<ul id="t1">
<li> Item 1 </li>
</ul>
```



Examples: Script Read Access

1. document.getElementById('t1').nodeName
2. document.getElementById('t1').nodeValue
3. document.getElementById('l1').nodeName
4. document.getElementById('l1').firstChild.nodeName
5. document.getElementById('l1').firstChild.nodeValue

Sample HTML

```
<ul id="t1">
<li id="l1"> Item 1 </li>
</ul>
```

Example	Returns
1	"ul"
2	"null"
3	"li"
4	"text" → (why?)
5	"Item 1"

Examples: Script Write Access

```
1. var list = document.getElementById('t1')
2. var newitem = document.createElement('li')
3. var newtext = document.createTextNode("Item 2")
4. list.appendChild(newitem)
5. newitem.appendChild(newtext)
```

Sample HTML

```
<ul id="t1">
<li> Item 1 </li>
</ul>
```

Example	Task
1	Sets list = the "ul" element
2	Creates newitem = new "li" element
3	Creates newtext = new "text" node
4	Puts newitem at the end of list
5	Puts newtext in newitem

Script: Stealing Browser History

- Script cannot directly access browsing history but …
  - If you have visited a website, the link color is different
  - Script can include an invisible link and check the color
  - Check to see if user visited competitors

```
var link= document.getElementById("myLink");
alert(window.getComputedStyle(link, null).getPropertyValue("color"))
```

```
function getLinkColor(url) {
  var a = document.createElement('a');
  a.href = a.textContent = url;
  document.body.appendChild(a);
  return document.defaultView.getComputedStyle(a, null).color;
}
```

Remote Script Code is Risky

- **Integrity**
  - Compromise your machine
  - Install malware rootkit
  - Transact on your accounts
- **Confidentiality**
  - Read your information
  - Steal passwords
  - Read your email

Security Solution

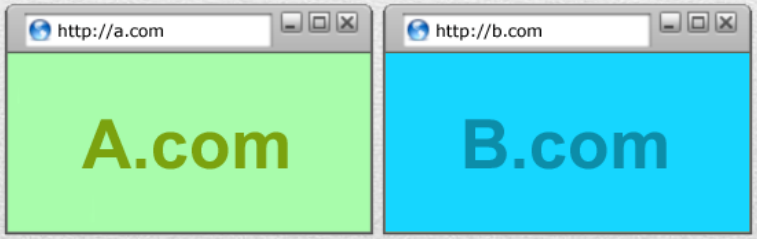
- **SandBox:**
  - Limited access to OS, network, and browser data.
  - Isolate sites in different security contexts.
  - Browser manages resources, like an OS.
  - ***Same Origin Policy:***
    - Can only read properties of documents and windows from the same server, protocol, and port
    - What if same server BUT unrelated websites/scripts?
      - *Does it matter where script refers to the same origin or NOT?!*
    - What if the server is vulnerable?
  - ***Privileges to signed scripts:***
    - Universal read, write, sendMail

Same-Origin Policy (SOP)

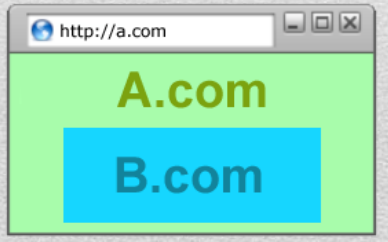
SOP: Goals

- Safe to visit an evil web site

- Safe to visit two pages at the same time



- Allow safe delegation



SOP: General Access

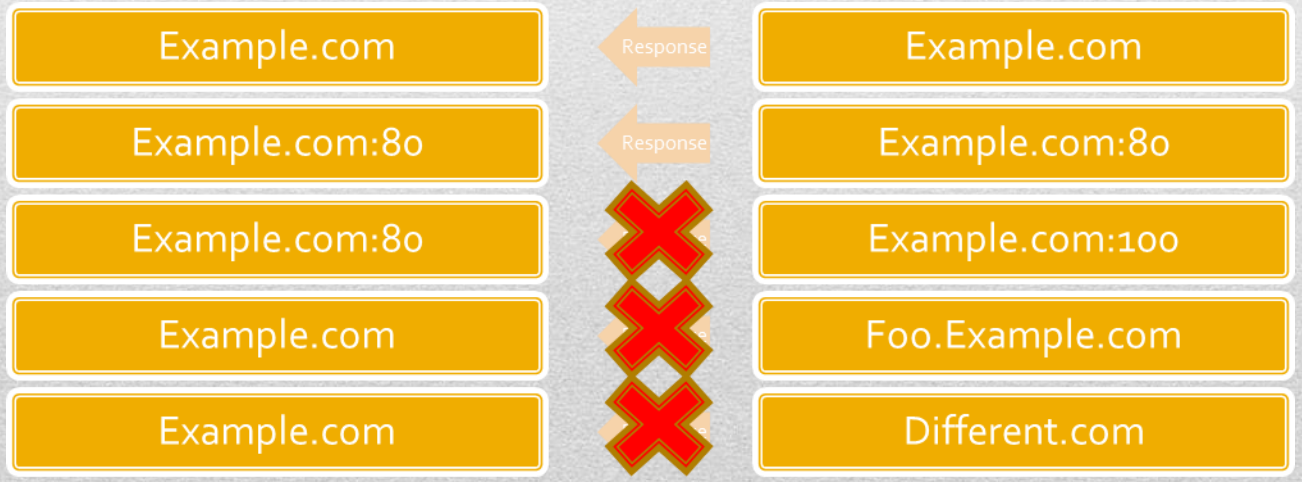
- Can only read properties of documents and windows from the same server, protocol, and port
  - Prevent Cross-site issues!
- Access from http://www.example.com/dir/test.html

Compared URL	Outcome	Reason
http://www.example.com/dir/page.html	Success	Same protocol and host
http://www.example.com/dir2/other.html	Success	Same protocol and host
http://www.example.com:81/dir2/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir2/other.html	Failure	Different protocol
http://en.example.com/dir2/other.html	Failure	Different host
http://example.com/dir2/other.html	Failure	Different host (exact match required)
http://v2.www.example.com/dir2/other.html	Failure	Different host (exact match required)

SOP: Get & Post Requests

- Get/Post ***request*** can be made from one domain to another.

- Get/Post ***response*** can only be read under these conditions:
  - If the ports match on both sites.
  - If the domain + subdomain match on both sites.



Post/Get Exception

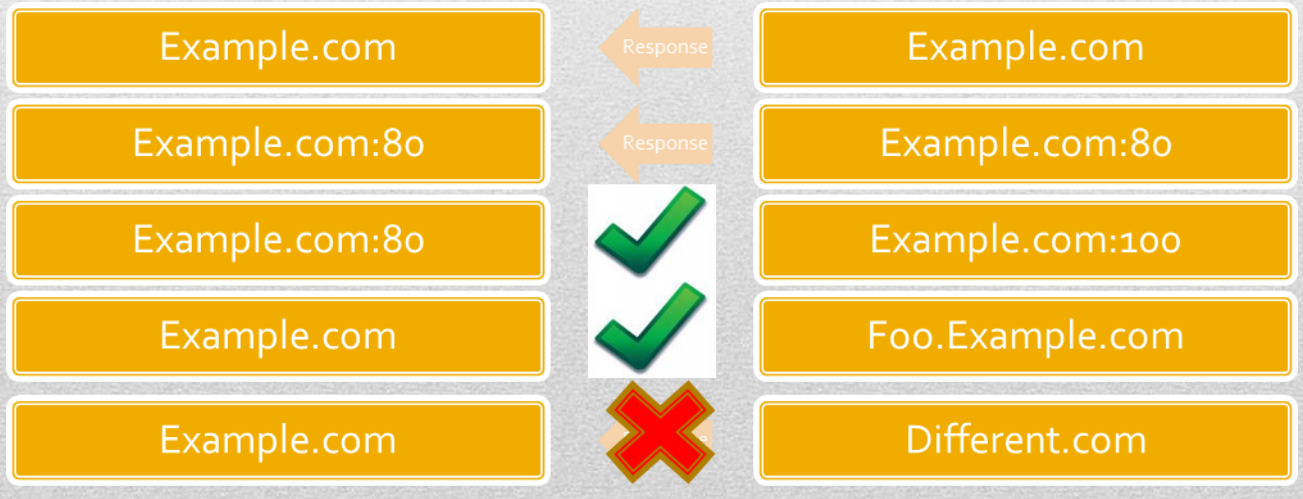
- Two different subdomains (different origin) are under the same domain & one is performing domain lowering (via document.domain)

- **Example:**
  1. **Clock.live.com** vs. **Vulnerable.live.com**
  2. **Clock.live.com** sets document.domain to **live.com**.
  3. **Vulnerable.live.com** is corrupted by attacker, s/he can set domain to live.com and access clock.live.com!

- **Threat:**
  - All eggs in one basket (*.google.com or *.live.com).
  - Cross-subdomain communication.
  - *Cross-Site Request Forgery*

SOP: Cookies

- Grant access if:
  - The domain is the same (Limited subdomain check)
    - Foo.bar.com → bar.com
    - bar.com → foo.bar.com
  - Does not respect port numbers!
  - Does not respect scheme/protocol:
    - Unless you opt in to secure attribute

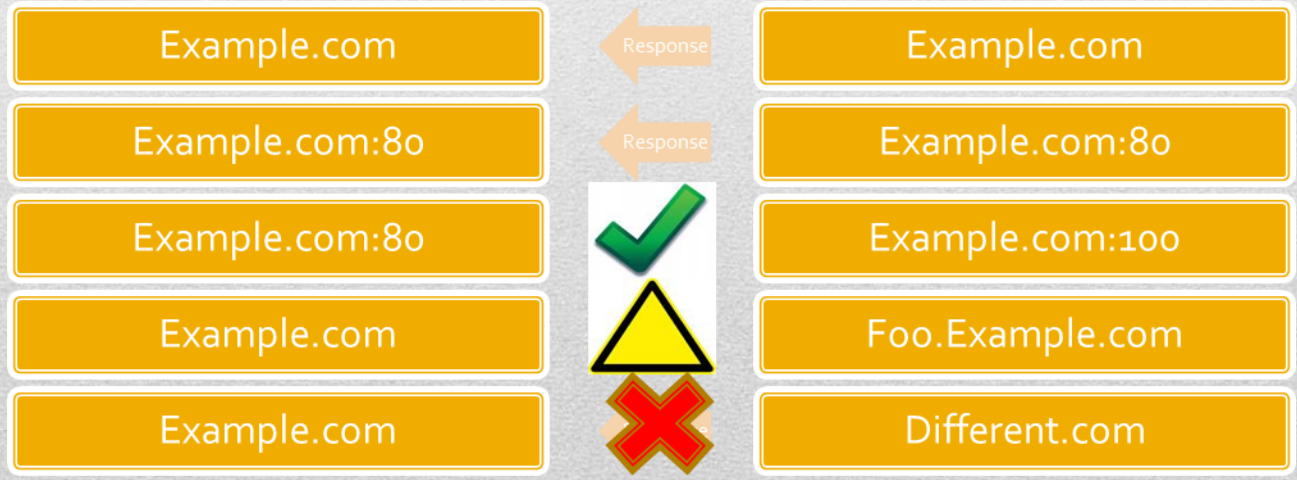


Risks

- Foo.bar.com can steal/poison cookie on bar.com.
- Foo.bar.com:1111 can steal/poison cookie on Foo.bar.com:2222.
- With regards to cookie and subdomains and ports are of limited security boundary.

SOP: IE

- Does not use ports during origin calculation.
  - You can read/write/script between:
    - Bar.com:80 and Bar.com:1234



Risks

- Host multiple web apps on different ports
  - Should be avoided when possible
- Ports are not a security boundary for IE
- Host web apps on separate domains

Network tools: NMAP and Wireshark

NMAP & Wireshark

- **Powerful open-source tools for:**
  - Network analyzing
  - Security check
  - Education

- **NMAP**:
  - Finds other alive systems and their open services/ports over the network.

- **Wireshark**:
  - Analyzes accessible interface packets: headers, statistics, etc.

NMAP

What is NMAP?

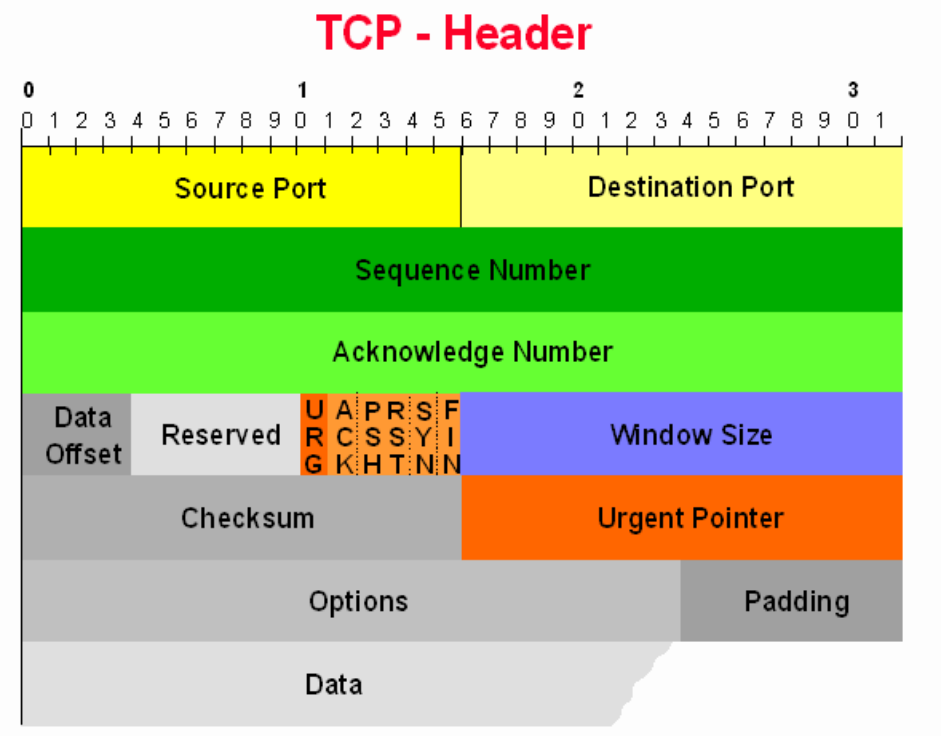
- **N**etwork **Map**per: Security scanner of hosts & services on a computer network.
- Written by Gordon Lyon (AKA Fyodor Vaskovich).
- Runs on different platforms.
- **Services:**
  - Host accessibility
  - Host OS
  - Open ports
  - Service names & versions
  - Estimated uptime
  - Device type
  - Presence of a firewall
  - Scriptable interaction

NMAP: Synopsis

- **Synopsis**

nmap [<*Scan Type*> ...] [<*Options*>] {<*target specification*>}

- **Scanning Protocols:**
  - TCP
  - UDP
  - ICMP
  - Ping:
    - ICMP ping
    - TCP ping



NMAP: Basic Port Scanning

- **UDP scan: -sU**
  - Sends 0-byte UDP packets to each port.
    - ICMP Port Unreachable → closed
    - Nothing → open
  - Slow due to ICMP frequency control [2^{16}!!] (but good on Windows ☺)
  - Useful for UDP services:
    - SNMP, NFS, the Back Orifice Trojan backdoor

- **TCP connect scan: -sT**
  - Begins TCP connection with SYN=1 on each port.
  - Follows SYN – ACK/SYN – SYN handshake.
  - Finds open/closed/filtered ports: (handshake / RST / ----)
  - Easy to detect by firewall or IDS.

- **SYN scan: -sS**
  - Follows SYN – ACK/SYN – RST to drop connection.
  - Typically does not appear in log files.

NMAP: Advanced Port Scanning

- Modern firewall and IDS can detect SYN (-sS) scan…
- There are other scan types to help.

- **FIN scan: -sF**
  - Sends packet with FIN=1.
  - Notes whether or not the connection succeeded.

- **XmassTree scan: -sX**
  - Sends packet with FIN=1, URG=1, PSH=1.

- **Null scan: -sN**
  - Sends packet no flags set.

NMAP: Host/IP Scanning

- **Ping scan: -sP**
  - First ICMP; if not responded, SYN or ACK.
  - Using **-P0** option removes ICMP ping.

- **IP protocols scan: -sO**
  - Sends a raw IP packet to each protocol.
    - ICMP Port Unreachable → not in use.
    - Nothing → in use.
  - ICMP frequency problem, but OK! [2^8]

NMPA: Other Scan Types

- **Idle Scan: -sI**
  - Needs IP spoofing a Zombie!
  - Gets information from the Zombie's IPID.
- **ACK Scan: -sA**
  - Sends ACK packet.
    - RST response → *unfiltered* OR Nothing → *filtered*.
  - If there is a firewall: can tell if stateless/stateful.
  - Usually used with other scan types
- **Window scan: -sW**
  - Sometimes infer open ports too!
- **RPC Scan: -sR**
  - Checks if a port is remote procedure call (RPC) service.
- **List Scan: -sL**
  - Just a list of IPs and DNS names.

NMAP: Basic Options

- **Verbosity: -v**
- **Version detection: -sV**
  - Service running on a port: product name & version.
- **OS detection: -O**
  - Gets OS information of the target.
- **OS & Version detection: -A**

NMAP: Other Options

- **Timing:**
  - From -T0 (paranoid > 5 mins) to -T5 (Insane ~ 0)
- **Decoy: -D**
- **Output logging:**
  - -oN (human), -oX (XML), -oG (Grepable)
- **Scan from a file: -iL**
- **IPV6: -6**
- **Stop & resume: Cntrl+C & -resume**
- **Fast scan: -F**
  - nmap_services ports
- **Setting TTL: -ttl**

NMAP: A few examples

- Ping scan of IPs: *which hosts are alive?*

  `nmap -sP 192.168.181.0/24`

  (specific range) `nmap -sP 192.168.181.90-100`

- Scan ports of an IP: *which ports are open*?

  `nmap -sS 192.168.181.102`

- Find OS of an IP: *what is its OS*?

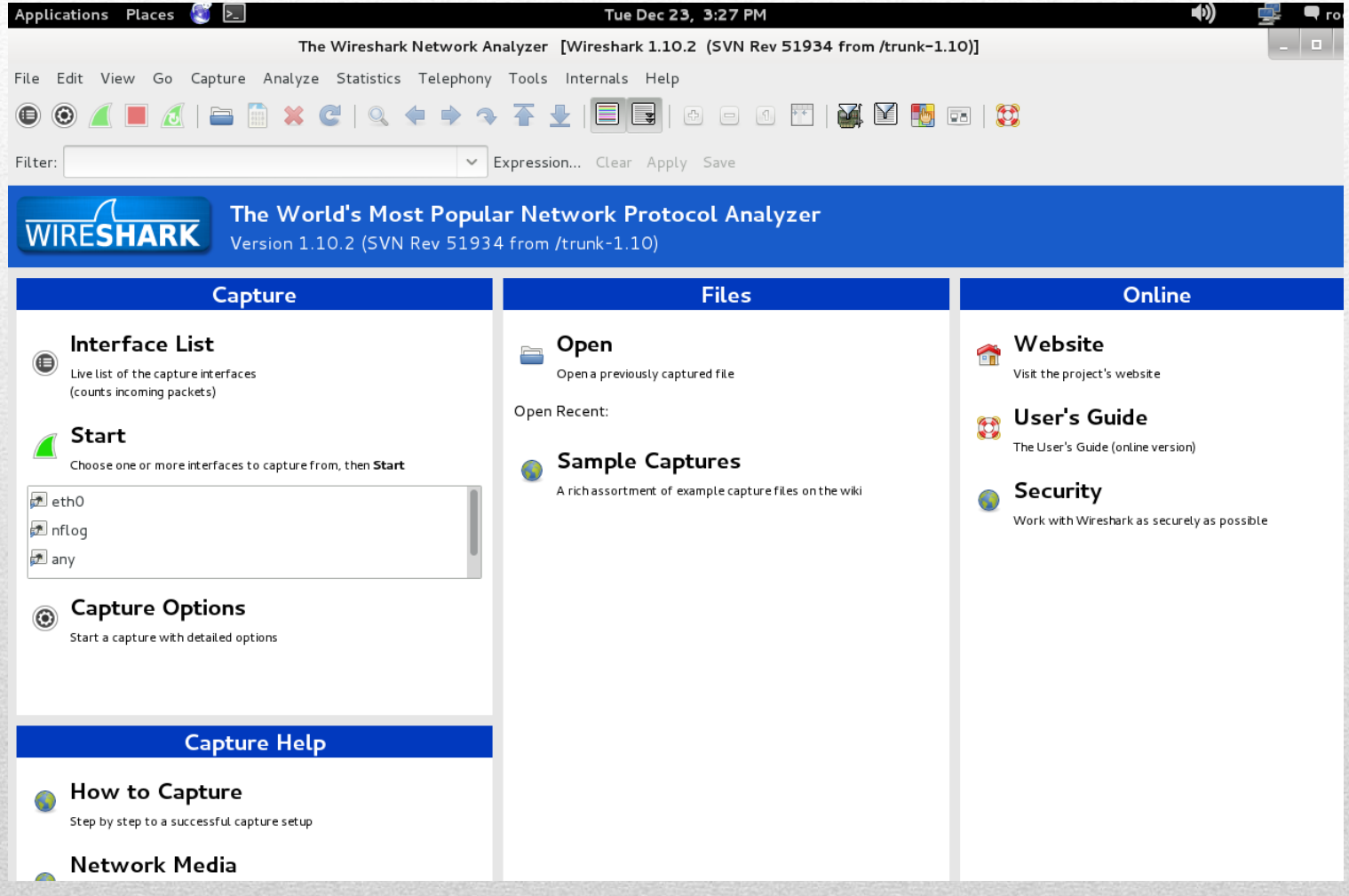
  `nmap -O 192.168.181.102`

Wireshark

What is Wireshark?

- **Open-source packet analyzer with GUI:**
  - For network troubleshooting & analysis.
  - The command-line version is called ***tshark***.
- Written by Gerald Combs: originally named *Ethereal*.
- Runs on different platforms.
- **Features:**
  - Data capturing: wire / network / USB traffic / file packets.
  - Ethernet, IEEE 802.11, PPP, and loopback.
  - Display filter.
  - Allowing for Plugins.
  - VoIP calls detection or even playing.

Wireshark: Graphical User Interface



Wireshark: Command Menus

- **File:**
  - Save & load captured packet data.
- **Edit:**
  - Find a packet, time reference or mark one or more packets,
- **View:**
  - Display of the captured data: e.g., colors, font, expanding and collapsing trees.
- **Go:**
  - Go to a specific packet.
- **Capture:**
  - Begin packet capture.
- **Analyze:**
  - Manipulate display filters, dissection of protocols, configure decodes.
- **Statistics:**
  - Summary of captured packets, protocol hierarchy statistics, etc.

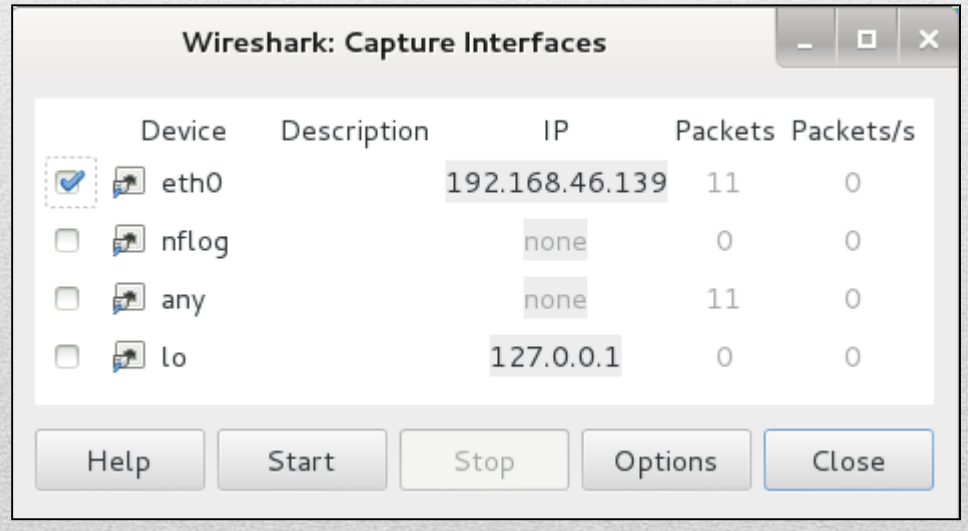
Wireshark: Network Data Capture

- **Notes:**
  - Root / Administrator privileges.
  - Right network ***interface*** to capture packet data from.
  - Capture right place in the network to see the traffic.

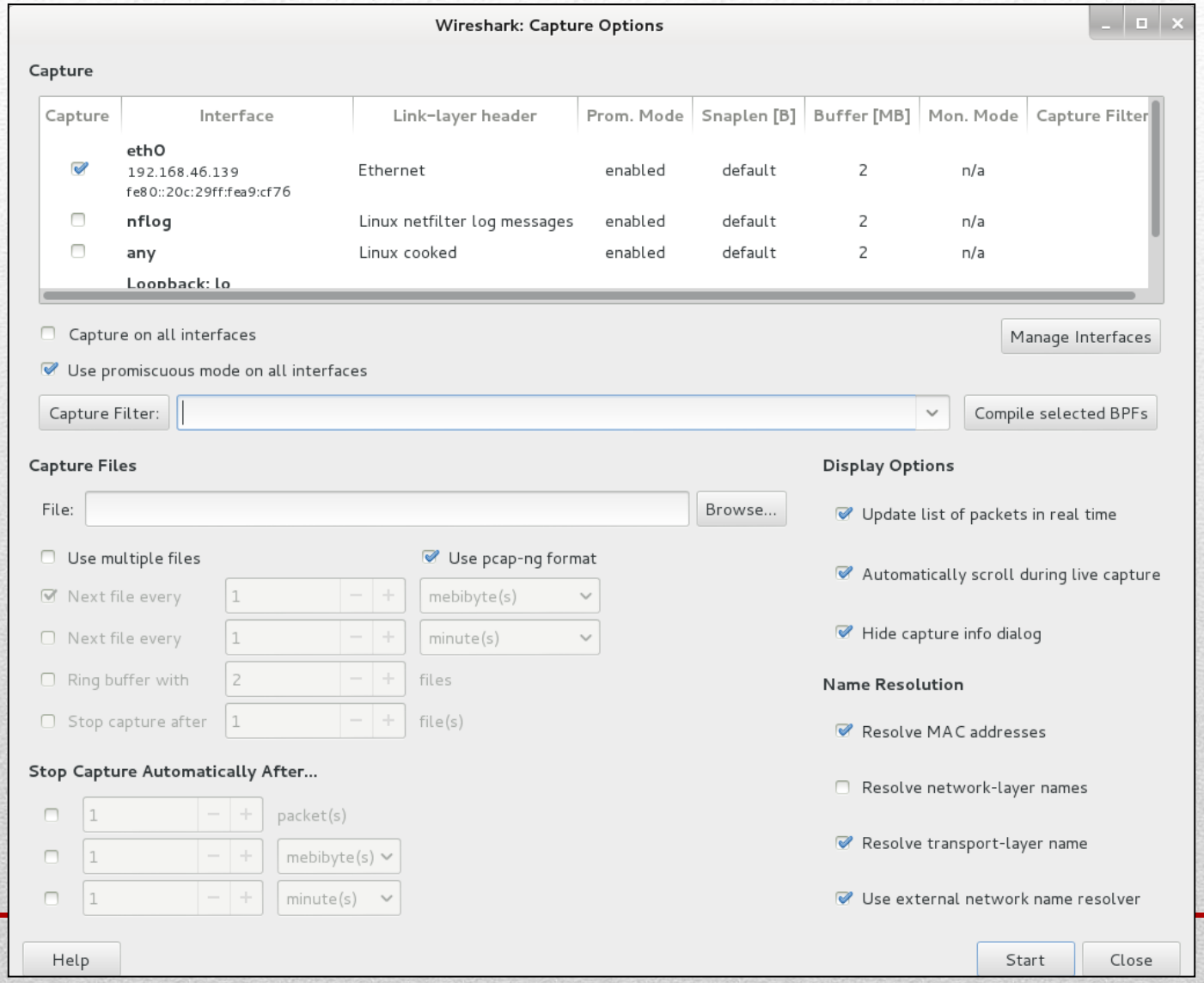
- **Four possible ways:**
  - Using the “Capture Interfaces” dialog box.
  - Using the “Capture Options” dialog box.
  - Immediately using “Capture Start” item (if options are set).
  - Command line:
    - wireshark -i eth0 -k

Wireshark: Capture Interface

- Capture menu → Interfaces
  - Possible to capture packets from multiple interfaces.
  - May Consume a lot of resource!

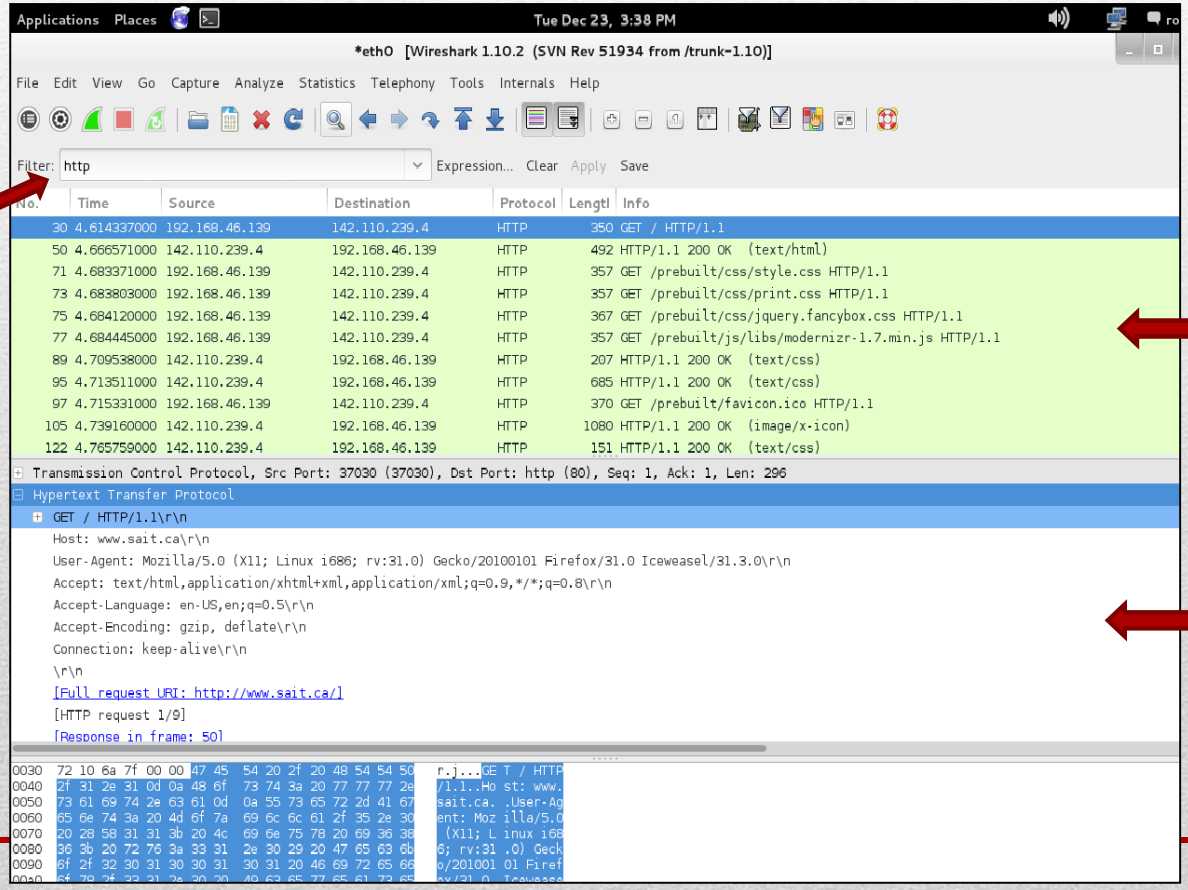


Wireshark: Capture Options



Wireshark: Capture Window

- Shows the packets of various types that are being captured.
- Stop the process when appropriate!



Show HTTP packets only

Captured packets

Contents of selected packet above