Computer Security
CS 426
Lecture 41

StuxNet, Cross Site Scripting & Cross Site Request Forgery

StuxNet: Overview

- Windows-based Worm
- First reported in June 2010, the general public aware of it only in July 2010
  - Have existed a year or more before being discovered
  - Updated several times
- Written to attack systems used to control and monitor industrial processes.
  - built by German firm Siemens.
- Seems to be a digital weapon
  - 60% (more than 62 thousand) of infected computers in Iran
  - Iran confirmed that nuclear program damaged by Stuxnet
- Origin is unknown, but conjectures exist

What StuxNet Does

- Its final goal is to reprogram industrial control systems (ICS) by modifying code on programmable logic controllers (PLCs) to make them work in a manner the attacker intended and to hide those changes from the operator of the equipment.

- Attacking three targets
  - Windows operating system
  - Siemens SIMATIC WinCC/Step 7 software that controls the PLCs
  - Siemons PLCs

Highlights

- Exploit four zero-day attacks
  - Zero-day exploit computer application vulnerabilities that are unknown to others or undisclosed to the software developer.
- Use stolen private key to sign rootkit drivers
- Code size: half a megabyte
- Written in multiple languages
- Updated over at least one year
- Likely by a nation-state.

How It Propagates?

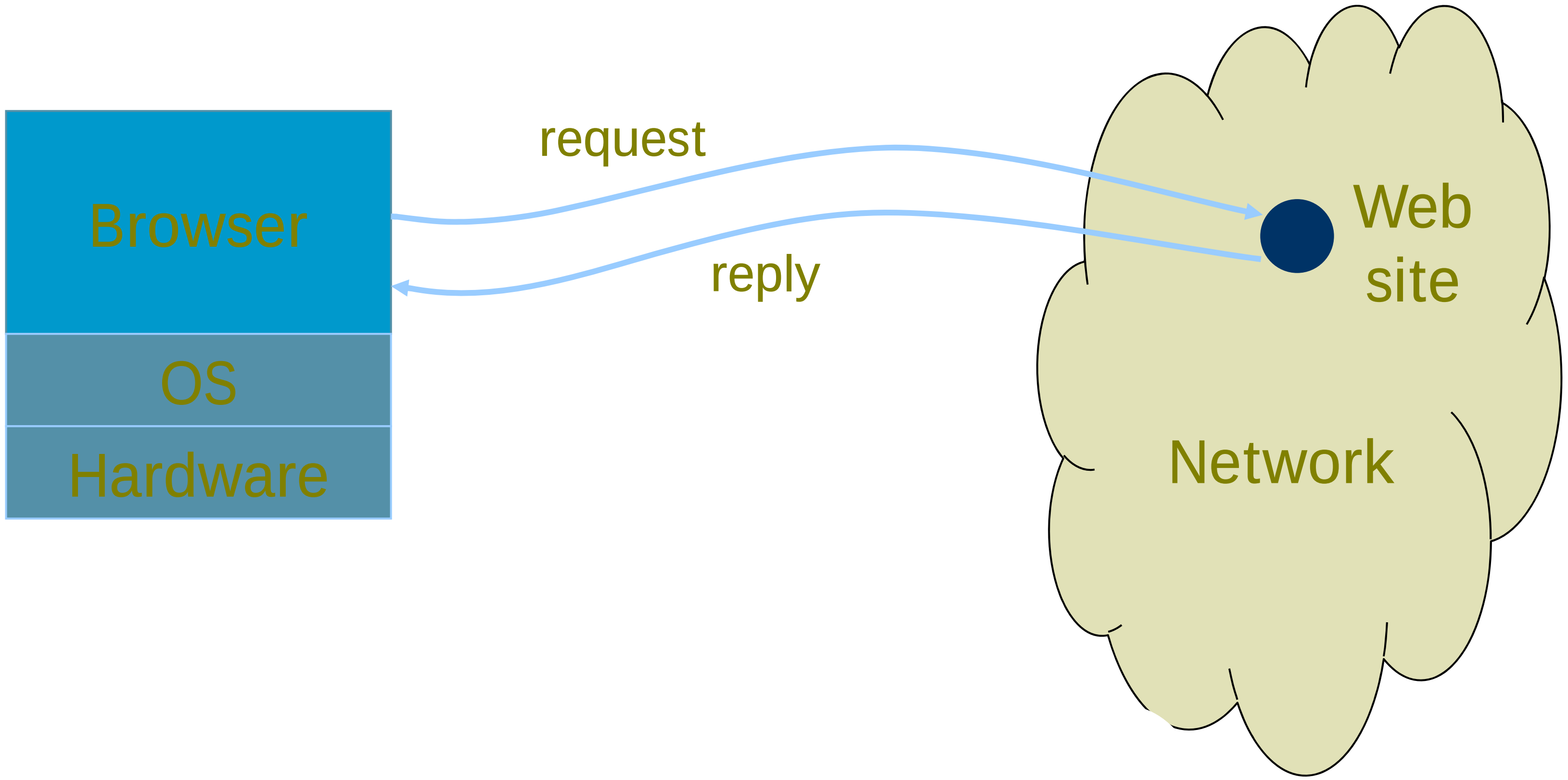
- Use USB drive to get on computers in closed networks
  - Exploit “Microsoft Windows Shortcut ‘LNK/PIF’ Files Automatic File Execution Vulnerability”
- Spread through local network through multiple methods
  - Exploit vulnerability in Windows Print Spooler
  - Exploit vulnerability in Windows Server Service (same as conficker worm)
  - Copy and execute itself through network shares
- Exploit two zero-day vulnerabilities for privilege escalation attacks on local computers
- Install a kernel rootkit on the computer
  - Kernel rootkite as device drivers digitally signed with stolen private keys from JMicron and Realtek.

How It Propagates?

- Copy itself into the control software Step 7 and make it loaded when the app starts
- Modify code on Siemons PLCs and hide its presence (rootkit for PLC)
- Use two websites for updates and information storage
- Also perform peer-to-peer updating

Cross Site Scripting

Browser and Network



- Browser sends requests
- Browser receives information, code
- Interaction susceptible to network attacks

Client Side Scripting

- Web pages (HTML) can embed dynamic contents (code) that can executed on the browser
- JavaScript
  - embedded in web pages and executed inside browser
- VBScript
  - similar to JavaScript, only for Windows
- Java applets
  - small pieces of Java bytecodes that execute in browsers

HTML and Scripting

```
<html>
    …
  <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>
```

Browser receives content, displays HTML and executes scripts

Scripts are Powerful

- Client-side scripting is powerful and flexible
  - host access
    - read / write local files
  - webpage resources
    - cookies
    - Domain Object Model (DOM) objects
      - steal private information
      - control what users see
      - impersonate the user

Same Origin Policy

- The basic security model enforced in the browser
- Web users visits multiple websites simultaneously
- SoP isolates the scripts and resources downloaded from different origin
  - bank.com vs. evil.org
- Origin = domain name + protocol + port
  - all three must be equal for origin to be considered the same

Same Original Policy: What it Controls

- Same-origin policy applies to the following accesses:
  - manipulating browser windows
  - URLs requested via the XmlHttpRequest
    - XmlHttpRequest is an API that can be used by web browser scripting languages to transfer XML and other text data to and from a web server using HTTP, by establishing an independent and asynchronous communication channel.
      - used by AJAX
  - manipulating frames (including inline frames)
  - manipulating documents (included using the object tag)
  - manipulating cookies

Problems with S-O Principle

- Poorly enforced on some browsers
  - Particularly older browsers
- Limitations if site hosts unrelated pages
  - Example: Web server often hosts sites for unrelated parties
    - http://www.example.com/account/
    - http://www.example.com/otheraccount/
  - Same-origin policy, allows script on one page to access properties of document from another
- Can be bypassed in Cross-Site-Scripting attacks

Cross Site Scripting (XSS)

- Recall the basics
  - scripts embedded in web pages run in browsers
  - scripts can access cookies
    - get private information
  - and manipulate DOM objects
    - controls what users see
  - scripts controlled by the same-origin policy
- Why would XSS occur
  - Web applications often take user inputs and use them as part of webpage (these inputs can have scripts)

How XSS Works on Online Blog

- Everyone can post comments, which will be displayed to everyone who view the post
- Attacker posts a malicious comment that includes scripts (which reads local authentication credentials and send of to the attacker)
- Anyone who view the post can have local authentication cookies stolen
- Web apps will check that posts do not include scripts, but the check sometimes fail.
- Bug in the web application. Attack happens in browser.

Effect of the Attack

- Attacker can execute arbitrary scripts in browser

- Can manipulate any DOM component on victim.com
  - Control links on page
  - Control form fields (e.g. password field) on this page and linked pages.

- Can infect other users: MySpace.com worm.

MySpace.com (Samy worm)

- Users can post HTML on their pages
  - MySpace.com ensures HTML contains no

    `<script>, <body>, onclick, <a href=javascript://>`
  - However, attacker find out that a way to include Javascript within CSS tags:

  ```
  <div style=“background:url(‘javascript:alert(1)’)”>
  ```

  And can hide `“javascript”` as `“java\nscript”`
- With careful javascript hacking:
  - Samy’s worm: infects anyone who visits an infected MySpace page … and adds Samy as a friend.
  - Samy had millions of friends within 24 hours.
- More info: http://namb.la/popular/tech.html

Avoiding XSS bugs (PHP)

- Main problem:
  - Input checking is difficult --- many ways to inject scripts into HTML.
- Preprocess input from user before echoing it
- PHP: **htmlspecialchars**(string)

  & → &amp;   " → &quot;   ' → &#039;

  < → &lt;   > → &gt;

  - **htmlspecialchars**(
    "<a href='test'>Test</a>", ENT_QUOTES);

    Outputs:

    &lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;

Avoiding XSS bugs (ASP.NET)

- ASP.NET 1.1:
  - **Server.HtmlEncode(string)**
    - Similar to PHP htmlspecialchars
  - validateRequest: (on by default)
    - Crashes page if finds <script> in POST data.
    - Looks for hardcoded list of patterns.
    - Can be disabled:

```
<%@  Page  validateRequest=“false"  %>
```

Cross site request forgery

Cross site request forgery (abbrev. CSRF or XSRF)

- Also known as **one click attack** or **session riding**
- Transmits unauthorized commands from a user who has logged in to a website to the website.

CSRF Explained

- Example:
  - User logs in to bank.com. Forgets to sign off.
  - Session cookie remains in browser state
  - Then user visits another site containing:

```
<form name=F action=http://bank.com/BillPay.php>
<input name=recipient value=badguy> …
<script> document.F.submit(); </script>
```

  - Browser sends user auth cookie with request
    - Transaction will be fulfilled

- Problem:
  - browser is a confused deputy

GMail Incidence: Jan 2007

- Google docs has a script that run a callback function, passing it your contact list as an object. The script presumably checks a cookie to ensure you are logged into a Google account before handing over the list.
- Unfortunately, it doesn’t check what page is making the request. So, if you are logged in on window 1, window 2 (an evil site) can make the function call and get the contact list as an object. Since you are logged in somewhere, your cookie is valid and the request goes through.

Real World CSRF Vulnerabilities

- Gmail
- NY Times
- ING Direct (4th largest saving bank in US)
- YouTube
- Various DSL Routers
- Purdue WebMail
- PEFCU
- Purdue CS Portal
- …

Prevention

- Server side:
  - use cookie + hidden fields to authenticate
    - hidden fields values need to be unpredictable and user-specific
  - requires the body of the POST request to contain cookies
- User side:
  - logging off one site before using others
  - selective sending of authentication tokens with requests

Coming Attractions …

- Final Exam