Brief View In Prioritizing Website Security

ROOTCON 6; Sept. 7-8, Presented by : nitrob

nitrob@synfyre.net

Disclaimer!

* The demos are done in a controlled environment and no other information will be disclosed that isn’t included or related in the presentation.

Overview

Hurr Durr *Introduction

@me

- First time speaker
    - please bare with me
- High School dropout
    - dropped out twice
    - haven't finished high school yet
- Former leader of PhilKer (Philippine Hackers)
    - 2009 to 2012
- Home brewed PHP Application Developer
    - made my first app when I was 15.
- Security Infrastructure Enthusiast
    - I'm insane in terms of implementing security.
- アニメオタク 「Anime Otaku」
    - I watch anime all the time, sometimes sub alongside with fansub groups

Notable Events & Happenings

* Things happened that are note worthy.

International Events

- 2011 is dubbed as “The Year of the Hacker”
  - Anonymous, Lulzsec, Rampant Indo-Pak Cyber Wars
- RSA Security, March ’11
  - Social Engineering crippled them easily
  - A target in Operations Aurora & Shady RAT
- PlayStation Network, April ‘11
  - Different websites with common SQL Injections
  - An estimated $171 million worth of loss.
- Citi Group, June ‘11
  - 200,000 customer account details stolen
- FBI Partner InfraGuard Atlanta, June ’11
- LinkedIn, June ‘12
  - 6.5 million password hashes stolen, 200,000 so far cracked.
- Hundreds more, that a worthy of being included but too many

Local Events

- OVP defaced over and over again from June’11~Jan’12
  - Done by : Philker, PrivateX
- FDA defaced, June 16’11
  - Done by : Philker
- PNRI defaced, August 25’11
  - Done by : PrivateX
- Chinese websites defaced, April’12
  - Joint by Anonymous #OccupyPhilippines, PrivateX, many more

Basic Information

* Information that will help you understand the basis of website security.

>101's

- First and foremost, never trust user input.
- Use the POST method for actions and GET for retrieval
- Be mindful of implementing security in all areas
- Design a system in an architect's perspective
- Be paranoid
- Think like an attacker would do
- Create logs for every important actions taking place
- Educate users and fellow colleagues in security
- Use HTTPS whenever possible

>XSS

- One of the most common attacking vectors
- Very common if a website mostly use $_GET
- Can be used by anyone with knowledge in exploiting HTML
- Persistent*stored / Non-Persistent*reflected

Preventing It

- Escape all possible HTML input
- Be aware of forms using URL parameters for processing
- Set cookies to “HttpOnly”

>LFI / RFI

- Only exists when a website uses file system combined with user input
- Attack vectors and severity of the damage may vary.
- Used to inject malicious code into a vulnerable website

Preventing It

- Be mindful of the usage of *require* or *include*
- Check first the included file if it exists
- Trimming the input by using basename() in PHP
- Set allow_url_include in the php config to "Off"

>CSRF

- IMO, rare kind of vulnerability in developed websites
- It relies on the authenticated user
- An indirect kind of attack whereas exploits the site’s trust

Limitations *courtesy to Wikipedia

1. The attacker must target either a site that doesn't check the referrer header (which is common) or a victim with a browser or plugin bug that allows referrer spoofing (which is rare).

2. The attacker must find a form submission at the target site, or a URL that has side effects, that does something (e.g., transfers money, or changes the victim's e-mail address or password).

3. The attacker must determine the right values for all the form's or URL's inputs; if any of them are required to be secret authentication values or IDs that the attacker can't guess, the attack will fail.

4. The attacker must lure the victim to a Web page with malicious code while the victim is logged in to the target site.

>>Preventing CSRF

- When necessary use the POST method when dealing with forms
- Double check data received into the server
- Apply additional checking to verify data

>SQLi

- THE MOST common attack method used
- Can be used by almost anyone even with beginner knowledge
- A lot of tools are available dedicated to deliver SQLi attacks
- One of the best reasons why developers get into trouble
- The severity of the attack may lead to different disasters

Preventing It

- Always escape input, it’s a simple rule but it will save you
- Limiting the number of queries that will execute
- Using proper coding standards

>RCE

- An advanced way of exploiting a vulnerable system
- Relies on executing code from a remote server to inject mcode.
- A combination of XSS/LFI/RFI/CSRF/SQLi can be executed

Preventing It

- All of what I've said above.

Diving Into Development

* My fundamentals into developing an application with extensive security to create peace and prosperity.

>From the Drawing Board

- Follow an application coding structure
- Design it like an architect
- Balance usability with security
- Apply user privileges/permissions
- Don’t mistake later for now
- Take part in the community

>>Follow An App. Coding Structure

- The best method is the MVC pattern
- If not, use apache's `mod_rewrite` to manipulate URLs
- Organize your scripts to prevent clutter
- Map out the entire application

>>Design It Like An Architect

- Start with the basics before going to the advanced stuff
- Look out for the tiniest mistake in development
- Take notes in every change made in development
- Consult wit the people who specializes in particular fields

>>Balance Usability With Security

- One of the essence of secured applications
- 50% logic & 50% security
- UI matters to some users

>>Apply User Privileges/Permissions

- In every area, there should be limits to a certain user/usergroup that can access

- Log certain activities that will help in maintaining security throughout the application

>>Don’t Mistake Later For Now

- You gotta do it, it helps later on

- You may regret it in the long run

- This will let you avoid disaster like law suits, user privacy concerns, stolen data, or even bankruptcy

>>Take Part In The Community

- Take part in security blogs to know the latest security trends and vulnerabilities.

- Let your community asses the security what you have now, and offer them bounties. As what Google, Facebook, & Microsoft would do.

- Organize hackthons that centers into developing your website for the greater good of the users who uses it.

Security By Design

* The fundamentals in creating security at the same time delivering what the application needs to function

>Security Design Examples

- 2 step verification
- Temporary/Application passwords
- User geography analyzer
- Advanced cookie implementation

>> 2 Step Verification

• To my knowledge, first implemented by Google Services in spring of 2011. Followed by Facebook months later; Went available for a few months in the Philippines.

• It registers two numbers, your mobile device number and your phone number as backup.

• Sends you a verification code, that will help the system verify you are currently logging in to the account

• Costly to implement for start up companies, but better for enterprise sized companies

>> Temporary/Application Passwords

• As seen on Facebook & Google (2 step verification)

• Provides the user to have a different password to be used on a 3rd party application (ex. Thunderbird & Tweetdeck)

• Temporary passwords provide extra security if a system or network you're trying to log on may be infected of keyloggers and trojans

• Expires on use

• Easy to implement, good for start ups who want extra security and recommended for enterprise sized companies having corporate financial data

>> User Geography Analyzer

• Facebook uses it, you may just never feel it.

• System will analyze automatically your past logins from the database containing data such as an IP address, browser useragent, etc.

• Compares your past logins if they are in the same geographical location. For example, if you frequently log in to city A with browser X, then someone logged in your account from city B with browser Y. The system will automatically notifies the user and lock the account temporarily.

• An advantage, but not recommended if your website is still small.

>> Advanced Cookie Implementation

- Once used it on SynFyre.
- A unique hash is tied to a cookie then cookie data on the database.
- Upon visit on the website, system will automatically looks up in the database for the data of the hash.
- Cookie data can be only be used in the server, therefore not exposed in client side.
- If some of the user data doesn’t match from the cookie data, user will be asked to input his/her password
- If the login is successful, updates the cookie data with the latest user data; If it fails, logs necessary details in the database

Improving Improvements

Improving what is already done to make it even more secure.

(LOL)

> Assessment Is Key

- Asses on what’s in development and what’s already in production and maintain communication with developer and security teams.

- Separate development and production assessment, and maintain a grip hold of what’s in production.

> Getting The Community Involved

• Let your users be the security ninjas by offering goodies to those who can find vulnerabilities first before someone wrecks havoc.

• It drives users to be good citizens instead of being bad, it also help the general community who uses it.

• This is the reason why Facebook & Google doesn’t get hacked. Since their users are also helping making them safe.

> If there’s something new, test it.

- Every now and then, when we release a new version of software to the public there’s always a chance that it will have a 0 day.
  - 0 days, are exploits that can be found on computer software upon it’s release without the acknowledgement of the public of such vulnerability.
- Even if it’s already secure test it again, blackhats always have a lot of tricks up their sleeve.

Conclusions & End Thoughts

Wrapping it all up

> Security Exploitation

- Tons of attack vectors and tons of tools available
- No system is relatively safe from vulnerabilities
- Don’t underestimate everyone
- Anything sent in HTTP can be forged and exploited easily

> Security Implementation

- Design it like an architect
- Balance the flow of development and security
- Don’t trust user input, filter everything
- Paranoia is included
- Use HTTPS whenever possible

> End Thoughts

• My contact details recently changed due to a new project, if you want to get a hold of me through email, loophaze@hakz.co

• Follow me on Twitter if you're interested in my blabber, @loophaze

• My state of mind in the hacker underworld is currently futile and will be back in the scene hopefully soon.

Question & Answer

Ask away!