Facebook Vulnerabilities

Vulnerability 01

Aurigma Image Uploader ActiveX control stack buffer overflows (https://www.cvedetails.com/cve/CVE-2008-0660/)

Description: ActiveX control which provides the Internet Explorer web browser to upload images called Aurigma Image Uploader. Numerous websites, including Facebook and MySpace, employ the Aurigma ImageUploader ActiveX control. In various attributes, including Action, ExtractExif, and ExtractIptc, this ActiveX component has multiple stack buffer overflows. Including version 5.0.30 and up to versions have been identified as potentially susceptible based on limited testing.

Impact: The outside, an unauthenticated attacker might be able to execute an arbitrary code with the user's privileges on a vulnerable system, by persuading a user to read a specially crafted HTML document (such as a web page or a HTML email message or attachment).

Solution(s):

- Apply an update.

Versions 5.1.0, 4.7.0, 5.0.41, and 4.6.31 each provide a fix for this problem. For further information, please visit the blog post from Aurigma. Microsoft Security Advisory (953839) also addresses this problem. Available through a variety of update channels, including Microsoft Update, this update sets kill bits for various Aurigma Image Uploader ActiveX control versions.

- Turned-off the Aurigma ImageUploader (ActiveX controls)in Internet Explorer.

After setting the kill bit for the CLSIDs, the ActiveX controls (Aurigma) could be turned-off in Internet Explorer. Be aware that not every version of the control may be included in this list. In Microsoft Support Document 240797, there are more facts on how to adjust the kill bit.

- Turned-off ActiveX.

ActiveX vulnerabilities appear to be prevented from being exploited by turning -off ActiveX controls in the Internet Zone (or any other zone utilized by a bad guy).The document "Securing Your Web Browser" provides instructions for turning off ActiveX in the Internet Zone.

Best Practices: Continually review issue reports for your web and application servers as well as other components of your Internet infrastructure. Update these products' fixes as needed. Use one or more of the popular scanners to periodically examine your website for buffer overflow vulnerabilities in your server products and your custom web applications. Developer must verify every line of code in the custom application that accepts input from users through a Hypertext Transfer Protocol (HTTP) request and make sure that it performs the necessary size checks on all such inputs. Even environments that are resistant to such attacks should nevertheless take this

precaution since excessively large inputs that are not caught may still result in denial of service or other operational issues**.**

Vulnerability 02

Facebook OAuth Framework (https://www.amolbaikar.com/facebook-oauth-framework-vulnerability/)

Description: OAuth 2.0 Authorization Protocol is used by the "Login with Facebook" feature to interchange tokens between Facebook.com and third-party websites. The vulnerability might provide attackers access to the OAuth pipeline and let them steal access tokens, which they could then use to hijack user accounts. The most popular apps' access tokens can be stolen simultaneously by malicious websites, which could also acquire access to various services and third-party websites. Like Tinder, Spotify, Netflix, Oculus, Instagram, etc.

Impact: When user visiting an attacker-controlled website could have had their first party access tokens for weak apps using Facebook's OAuth flow stolen as a result of an erroneous post message configuration.

Solution(s):

Validating Inadequate Mitigation & Bypass

- Deprecated is the "/connect/ping endpoint". It can no longer be used to create access_token for any applications permanently.
- To stop the JS execution in page_proxy, the __d("JSSDKConfig") line was introduced to XD_Arbiter.

Instead of creating closed_window and postMessage() for client origin, www.facebook.com ignores the redirect status to xd_arbiter. causing the attack to fail. For Chrome, the "m","mobile","touch" etc. rules are the same, but not for Firefox. You may be familiar with how Facebook interacts with User-Agent and subdomains.

All browsers responded to the HTTP 302 redirect header when you entered the "mbasic.facbook.com" site.

https://mbasic.facebook.com/dialog/oauth?client_id=124024574287414&redirect_uri=https%3A%2F%2Fstaticxx.facebook.com%2Fconnect%2Fxd_arbiter%2Fr%2F7SWBAvHenEn.js%3Fversion%3D42%23origin%3Dhttps%253A%252F%252Fwww.instagram.com%252F

- It is no longer permitted to alter or meddle with xd_arbiter. (Only accept absolute file paths, like "xd_arbiter.php")
- Resource file "7SWBAvHenEn.js" was deleted from the server.
- Regex validation filter has been added to another JS resource.

Best Practices: Always take the technology into account while implementing OAuth, as well as whether the application is client-side or server-side, which can keep secrets private or not. Use the PKCE Authorization Code flow wherever possible. Machine-to-machine flows might be an exception. To transfer secrets, use header values or POST arguments. Put additional security headers like a Referrer-Policy in place when there are no other options (for instance, in older systems that cannot be moved).

Vulnerability 03

HTTP Desync Attack (Request Smuggling) / Session Hijacking (https://github.com/AnkitCuriosity/WriteUps/blob/main/HTTP%20Desync%20Attack%20(Request%20Smuggling).md)

Description: The front-end server was found to be using the Content-Length header, while the back-end server was found to be using the Transfer-Encoding header to determine the length of HTTP request. This desynchronization in determining the length of the request between the servers could be abused to escalate a Mass Account Takeover scenario by using the POST parameters of URL encoded form to log requests from legitimate users.

Impact: logging legitimate user requests using the POST parameters of a URL-encoded form.

Solution(s):

- Implements thorough input validation and sanitization: To prevent the server from processing dangerous content, it is crucial to make sure that all input received from clients is correctly validated and sanitized.
- Implement rate restriction: Rate limiting can assist stop an attacker from sending a lot of requests to the server at once.
- Make use of web application firewalls (WAFs): A WAF can thwart HTTP desynchronization attacks by identifying and preventing nefarious requests.
- Make use of content security policies (CSPs): CSPs can help avoid injection attacks by outlining the kinds of content that the web application is permitted to load.
- Implement correct error handling: To stop sensitive information from being exposed to attackers, it's critical that your web application handles errors and exceptions effectively.
- Maintain the most recent security patches on your web application and servers: To keep your web application and servers safe from the newest dangers, it's crucial to consistently apply security patches and updates to them.
- Educate your developers: It's crucial to make sure that your developers are aware of the dangers posed by HTTP desynchronization assaults and request smuggling. They also need to be made aware of the necessity of using secure coding techniques to fend off these kinds of attacks.

Best Practices: By changing the front-end server's configuration to normalize ambiguous requests before forwarding them, specific occurrences of this vulnerability can be fixed. Cloud flare and Fastly appear to successfully implement it, making it the only practical option for CDNs who don't want to expose their clients to risk.

Back-end servers must reject ambiguous requests outright and cut off the connection in question; normalizing requests is not an option. So advise concentrating on preventing request smuggling via the front-end server rather than rejecting requests because doing so is more likely to impact valid traffic than merely normalizing them.

Vulnerability 04

Information Exposure (https://github.com/arjun334/Facebook_Vul/blob/master/Facebook.pdf)

Description: The input text box on the Facebook app has both a username and password. Because the username property is labelled as "not password" on the keypad input text box, it can be snooped by accessibility services. The user can select between showing the attribute "password" and "not password" in the password field's show/hide option.

Impact: When a user hits the "show password" button, the attribute changes to "not password," allowing a trojan to sniff via the accessibility service.

Solution(s): The password field's Show/Hide button should be removed.

Best Practices:

- Classify the information that the application processes, stores, or transmits. Determine which data is sensitive in line with privacy laws, regulatory requirements, or company requirements.
- Utilize controls according to the classification.
- Avoid keeping private information around for too long. Use tokenization or even truncation that complies with PCI DSS or dispose of it as quickly as possible. Unrestrained data is impossible to steal.
- Be sure to encrypt all sensitive data while it is in transit.
- Ensure that standard algorithms, protocols, and keys are up-to-date, robust, and properly managed.
- Utilize protected (secure) protocols like TLS with cipher prioritization by the server, perfect forward secrecy (PFS) ciphers, and safe parameters to encrypt all data in transit. Utilize Hypertext Transfer Protocol Strict Transport Security (HSTS) directives to enforce encryption.
- Remove caching from responses containing sensitive data.

- Use powerful adaptive and salted hashing algorithms, such as Argon2, scrypt, bcrypt, or PBKDF2, to store passwords.
- Independently confirm the efficacy of settings and configuration.

Vulnerability 05

Privacy Violation (https://github.com/arjun334/Facebook_Vul/blob/master/Facebook.pdf)

Description: The software is inadequate to stop actors who are either (1) not explicitly permitted to access the data or (2) are not acting with the associated parties' implied authorization from accessing private information (such as credit card numbers).

Impact: All the personal information on the input text box of the keypad is labelled as "not password" when a user creates a new Facebook account via the Facebook program, making it sniffable by accessibility services.

Solution(s): Information that should be kept private from other apps should be.

Best Practices:

- Coordinate the Privacy and Security Teams: The opinions of these two teams frequently diverge: The privacy team is particularly interested in understanding the types of data collected, how it is kept, and when and how it should be deleted. Privacy is concerned with knowing what kind of data is being used, as opposed to security, which is concerned with protecting the collected data.
- Organizations Can Manage Change Better with a Foundational Privacy Practice: Having strong policies, processes, and tools in place to manage data protection and breach notification requirements is one aspect of having a solid privacy practice.
- Adopt Technology that Scales and Adapts to Your Needs: Organizations are using orchestration and automation tools to help create a platform for uniform, repeatable processes for privacy and security teams in an effort to speed up response times to privacy breach reports.