ADVANCED USES OF CSRF

The following advanced techniques that use CSRF (Cross-Site Request Forgery Attacks) attacks have been observed in recent years.

Bypassing CSRF protections with click-jacking

This recently-evolved technique can be used to bypass CSRF protection and submit POST method-based forms with attacker controlled data, using click-jacking. (See the highlight box on click-jacking for more information.) The best example of this attack is exploiting email update services. Such services are quite common in Web applications. In this, the attacker manages to force victims to update their e-mail IDs with that of the attacker, so that the attacker can then compromise the victims' account by performing a password reset.

This attack can occur even if the Web application contains tokens for CSRF protection. A elaborate description of this is available on this blog post.

Click-jacking is an attack involving embedded objects on a maliciously crafted Web page. Using framed content, or that from Flash, Silverlight, or Java, the attacker places a transparent or invisible click button beneath the mouse, so that whenever the user clicks on something they see on the page, the user is also clicking to an unseen website that may contain malicious code. The attack can also take advantage of dynamic HTML and CSS (Cascading Style Sheets) code for further disguise.

The difference between CSRF and click-jacking is that in CSRF, the victim's browser performs the attack (loading the state-changing URL directly) without the victim clicking to launch it, while in click-jacking, the user actually interacts with something, but the action is "hijacked" by placing a layer between the user and the page element that launches a legitimate action.

Using XSS to bypass CSRF protection

This technique applies to websites that have an application that guards against CSRF, yet have pages that are vulnerable to XSS attacks. It’s a big misconception that guarding against CSRF will also secure the application against XSS. Using XSS, attackers can bypass the CSRF protection, and can automate any action that can be done on the application, without problems. Attackers simply read the site-generated token from the response, and include that token with a forged request. One such example of this is attackers exploiting an XSS vulnerability of a website to add a fake user with administrator privileges, when the site is secured against CSRF attacks.

A good discussion on this type of attack (with code) is available in the *Further Reading* section at the end of the article. Please do spend some time on it. It was by using an XSS vulnerability that the Samy worm bypassed MySpace’s CSRF protection in 2005, infecting over one million accounts within just 20 hours of its release.

Attacking intranets

Exploiting intranets involves CSRF attacks through IP-based implicit authentication schemes. Most intranet Web servers are vulnerable to this type of attack. The fact that many intranet servers continue to use default passwords, leave hosts unpatched, and blindly rely on perimeter firewalls to block external attacks has given rise to intranet vulnerabilities to CSRF and malicious JavaScript. This malicious JavaScript, once behind the firewall, can attack the intranet; since it is independent of the operating system and the Web browser, it is hard to defend against.

One such example is port scanning of the intranet Web server, using JavaScript and CSRF. The JavaScript constructs a local (intranet) URL that contains the IP address and the port to be scanned. The script then includes an element in the Web page (such as an image, IFRAME or remote script) that is addressed by the URL.

Also, using JavaScript time-out functions, and event handlers like OnLoad and OnError, the script can decide whether the host exists and whether the given port is open. If a time-out occurs, the host probably does not exist. An OnLoad event indicates that the host probably runs a Web server, and an OnErrorevent indicates that the host exists, but the port is closed. The typical attack scenario is shown in Figure 6.

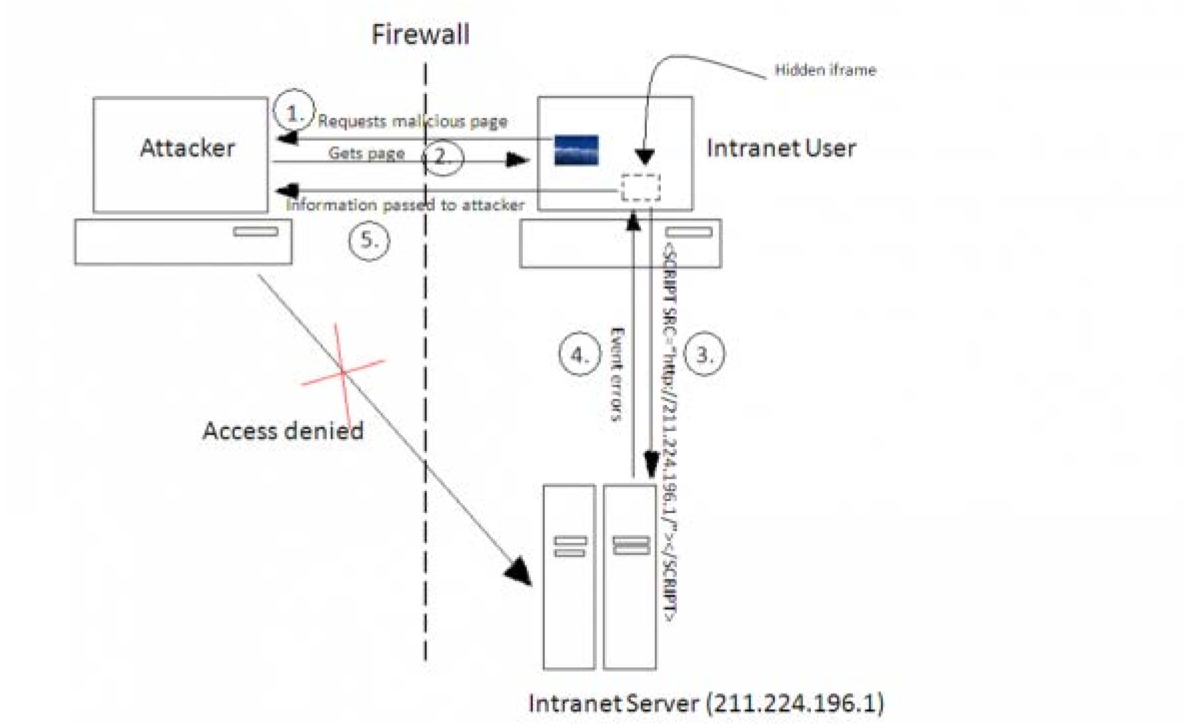

Figure 6: Port scanning intranet server

In this scenario, the attacker forces/convinces the intranet user to visit a malicious page on the attacker’s server, which contains the following HTML in a hidden IFRAME:

```
<SCRIPT SRC="http://211.224.196.1/"></SCRIPT>
```

As soon as the user executes this page, the information about the existence of the Web server is sent to the attacker in the form of errors. Apart from this, the attacker can fingerprint applications/routers/network devices, and also might locate HTTPS or development servers by varying the ports.

Source : http://www.opensourceforu.com/2010/11/securing-apache-part-3-xsrf-csrf/