My name is Brennan Brazeau and I am a member of the Security Systems Ethical Hacking team. In this video I will be talking about one of the more common weaknesses found in insecure code: Cross-Site Request Forgery.

This type of vulnerability is currently ranked #8 on the OWASP Top 10 chart and is a very commonly exploited vulnerability type.

What is a Cross-Site Request Forgery vulnerability?

Cross-Site Request Forgery is a web application vulnerability that makes it possible for an attacker to force a user to unknowingly perform actions while they are logged into an application.

Attackers commonly use CSRF attacks to target sites such as cloud storage, social media, banking and on-line shopping, because of the user information and actions available in these applications.

According to a recent IBM X-Force Threat Intelligence report, approximately 23% of all applications tested are vulnerable to Cross-Site Request Forgery.

Why should you be concerned about CSRF?

Depending on the action being performed by a user, a CSRF vulnerability can have serious consequences for a user using the web application.

The main issue is that users are typically unaware that malicious actions are being performed by an attacker. This means that changes can be performed in the name of the user without their knowledge or approval.

Practical applications of CSRF by an attacker can range from embarrassing social media posts to money being stolen from your online accounts.

Online Banking CSRF â€“ How is this accomplished?

A standard CSRF attack would begins as follows.

While logged into your banking application, you visit a page that contains a CSRF attack.

In this page, a hidden automatic request is planted and is used to transfer money from one account TO another.

The attacker now has the userâ€™s money and there is no indication that anything untoward has happened.

In this demo, I will show how a CSRF attack can occur against a banking application.

First, I will log into my banking application to perform some day-to-day banking transactions.

I will then open a new tab and go to a site I frequent. Thereâ€™s a new update with some cute cat pictures. Letâ€™s click on that link.

Did you see that? If we look at our recent transactions, we can see that $5000 has been removed from our account.

Letâ€™s look at the traffic we were recording.

A proxy program was setup to record all of our traffic so we can see what requests have been sent.

Unbeknownst to us, upon visiting this cute cat link, a request was sent to our bank to transfer $5000 from our account to someone elseâ€™s account.

Those cute cats have cost us quite a bit of money!

How come this was allowed to occur?

This attack was successful for 2 main reasons:

The banking application allows for a request to originate from servers other than itself.

The request used does not have a unique token that is tied to the userâ€™s session. If this token was absent or incorrect, it would stop the transaction from being performed.

To help find these issues, you can used a tool like AppScan STANDARD to test for CSRF issues in your application.

Upon selecting only the CSRF tests in the test policy and performing a full scan, we can see that our banking application contains a number of CSRF issues that should be fixed.

This concludes our coverage of the 8th OWASP Top 10 category. I hope you found this information useful. I wish you best of luck in writing and maintaining secure software!