

## Cross Site Request Forgery (CSRF) – an Interview with Amichai Shulman CTO and Co-founder of Imperva

Listen to Podcast [here](#).

**Brian Contos:** Welcome to the Imperva Security podcast. I'm Brian Contos, chief security strategist for Imperva. If you would like to learn more about this subject and Imperva, visit [Imperva.com](http://Imperva.com), check out our [blog](#), follow us on [Twitter](#), or send us an e-mail at [blog@imperva.com](mailto:blog@imperva.com).

Welcome to the show, Amichai.

**Amichai Shulman:** Thank you, Brian. It's great to be here again.

**Brian Contos:** Amichai, on this podcast we're going to be talking about CSRF. What exactly is CSRF?

**Amichai Shulman:** CSRF is a very simple attack. We have been discussing here hijacking and other types of attacks that are more complex to mount and CSRF is very simple. CSRF is where you have a link to a sensitive application, some transaction in that application and embedded within an attacker's controlled page. It works mostly because of careless users and because of tab browsers. You go to your browser, you open your banking application, your online banking application. You log in. You do whatever you do and then you leave this tab open with the application and then open another tab on the same browser.

And go wherever you feel like, and use sites, portals and so on. Quite a few of them are not as secure as we would like them to be, obviously.

Then if you get to a page that has been compromised or is controlled by a malicious individual to begin with, and that page contains a link that instructs the banking application to do something like fund transfer or some kind of financial transaction, then that request would go to the banking application with the session identifier that you have already created by logging into the application.

The transaction will be automatically executed without you noticing it. That's as simple as cross site request forgery is embedding a link within a page owned by the attacker. The link would be an image item within that page. It could be a scrape reference within the page. It could be everything that accepts a URL reference within an HTML page.

**Brian Contos:** It seems like an incredibly simple attack. We've been of course discussing some rather complicated ones over the last few weeks. Why is it that we're not hearing about this more or seeing more cases where this type of thing has been exploited?

**Amichai Shulman:** You know, it really puzzles me, Brian, because one thing that I can say for sure is that most applications out there today are indeed vulnerable to cross site request forgery. I would say at least the same number of applications that are vulnerable to other types of attacks such as SQL injection. It is probably more so, because if you have something that is really hard to plan for within your application, unless you are really thinking hacking, you're not going to put in safeguards. So most applications are vulnerable.

## CSRF with Amichai Shulman

We have seen actual vulnerabilities out there. Google had to be under attack with its Gmail service and quite a few times cross site request forgery vulnerabilities where indeed discovered in parts of the service.

We have recently seen a publication by a security researcher who demonstrated how cross site request forgery can be used with another well known application. Yet, the actual public incidence is really scarce.

We know of one rather large incident in a banking application in Korea. We know of another incident in which cross site request forgery through the Internet was used to attack internal routers through their web administration interface but that is it pretty much.

You want my honest opinion? I think that this is because cross site request forgery is the type of breach that would actually yield damage to the application or to the victim but is not represented through information leakage. So you don't have to report it. Therefore, most of the incidents in my opinion today are kept private.

**Brian Contos:** Interesting so it's actually a question of disclosure potentially more so than the fact that these attacks are actually going on.

**Amichai Shulman:** I think that's the issue here. It's a matter of disclosure. The simple matter of how can you detect cross site request forgery attacks it's very difficult. If you have SQL injections and no one would say yes, everybody would say no we're not seeing any SQL injection attacks. That's because nobody knew how to look for them. And I think that with cross site request forgery today, it's a matter of disclosure and matter of knowing how to detect them.

**Brian Contos:** So, Amichai, my experience has been that often the attacks that are the simplest can sometimes be the most difficult to be prevent. What are some of the risk mitigation steps that can be taken for CSRF?

**Amichai Shulman:** Well, you're right on the point, Brian, because it's so simple, it's so inherent to how applications work that it's never easy to mitigate. Ideally, what you would like to do against cross site request forgery attacks is to have a random, hidden field with each and every form or transaction in your application. That random, hidden parameter would be generated at least per user session. And if you have that in place, then an attacker cannot embed actual, working links in his own page. Now, as we all know, going back and fixing, rewriting, recoding each and every piece of your application is not always feasible. It's a lot of work. It costs a lot of money. It takes a lot of time. And sometimes you just don't have the source code at your disposal. So, I think that people should consider alternatives.

One alternative is to evaluate other parts of the HTTP request, such as the HTTP referrer header. And it turns out that most of the forms and transaction requests within your application are accessed through other pages of the same application, so you would expect the HTTP referrer header to reflect that fact. If you see a request for such a transaction, or a form that is submitted with the referrer HTTP header field indicating an external application, an external page, then that, in most cases, would be the result of cross-site request forgery.

So, at least for your sensitive transactions, sensitive forms, I would probably recommend using this type of mitigation. Which is not perfect, because researchers have shown that in a number of instances, it can be bypassed. It's not perfect, because there are scenarios,

such as mashups, in which you would like to allow this kind of behavior. But it's very easy to implement, and considering the 80/20 rule of thumb, it really gives you good protection.

Other alternatives. Again, in some cases, you just want to have your potential victim aware of the fact that this may be fraud. So, if you detect this kind of referrer discrepancy, you can issue a CAPTCHA, something that requires human response and human intervention, so the potential victim can then become aware of any activity that should not be happening.

And of course, I think that working with today's advanced AJAX frameworks, one, can actually embed the random numbers that I've mentioned earlier inside the engine code for the framework. And this can actually be done without recoding the application, but with external solutions such as web-application firewalls.

**Brian Contos:** Well, Amichai, we have just enough time for your closing thoughts on CSRF. Any last bits of advice you could give our listeners?

**Amichai Shulman:** Yeah, certainly. You should be careful. We've talked about this at the beginning of the year, in our yearly trends webinar. And I think that cross-site request forgery is going to be one of the major vehicles for attackers in the next few years, much more than cross-site scripting, for example. Most applications are inherently vulnerable to it. It's very easy to deploy. We're starting to see all the pieces of the attack coming together. One of them, for example, is making sure that when the victim is visiting the attacker's page, that victim is already logged in to the target application. And we're going to actually show how Google can be very helpful in such a scenario for the attacker.

So all the pieces are coming together that can make it very simple. Most applications are vulnerable. We're going to see much more of cross-site request forgery in the next year or two.

**Brian Contos:** Very good advice. And of course, just because the attack is more Columbo than James Bond doesn't mean that you don't want some focus an effort on it. Amichai, as always, thanks for joining us.

**Amichai Shulman:** Thank you, Brian.

**Brian Contos:** If you would like to learn more about this subject and Imperva, visit [Imperva.com](http://Imperva.com), check out our [blog](#), follow us on [Twitter](#), or send us an e-mail at [blog@imperva.com](mailto:blog@imperva.com).



North America Headquarters  
3400 Bridge Parkway  
Suite 101  
Redwood Shores, CA 94065  
Tel: +1-650-345-9000  
Fax: +1-650-345-9004

International Headquarters  
125 Menachem Begin Street  
Tel Aviv 67010  
Israel  
Tel: +972-3-684-0100  
Fax: +972-3-684-0200