

Clickjacking – 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, check out our [blog](#), follow us on [Twitter](#), or send us an e-mail at blog@imperva.com.

Brian Contos: Joining me today is Amichai Shulman, co-founder and CTO of Imperva. Welcome to the show, Amichai.

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

Brian Contos: Amichai, today we're talking about clickjacking. This term has been all over the media, but for those listeners who maybe need a refresher, could you give us a little bit of background? Exactly what is clickjacking?

Amichai Shulman: Sure, Brian. You're right, this is one of the most hyped buzzwords in terms of application security recently. I found out that it's very difficult to say what clickjacking is until you see it. Once you see it, it immediately strikes you what it is. In essence, it's having users commit transactions in sensitive applications without their knowledge, by covering the actual application with pages that are taken from a server controlled by the attacker. The entire attack is based on people using a multi-tab browser in a way that one tab is open with their sensitive application, and another tab would be used for general surfing.

Once the user's logged in the application in one tab, there's already a session established between that browser -- regardless of the tab you are in -- and the sensitive application. Now when the user is browsing through a different tab to a general site on the web, that site, that page, could be controlled by malicious individuals.

In fact, there's a method in which a malicious page can actually open a page from the sensitive application in an invisible frame, and on top of that page just lay out a completely different GUI in a manner that some of the fields and buttons from the original sensitive application page can be thought to be part of the new page.

When you click on buttons and when you fill forms in the fake page -- which looks unrelated to the sensitive application in any manner -- you actually do stuff, or commit transactions, on your sensitive application.

So, for example, an attacker could just present an innocent-looking page with a couple of poll questions, a question that looks like, "How [much], in your opinion, would the product in the following picture cost?"

You have a form to type your numeric answer and then click "OK." In fact, this page is overlaying a fund transfer page from a banking application to which the user is already

Clickjacking with Amichai Shulman

logged in. Actually, you put in an amount which will be taken from your account and transferred to a different account.

This is basically what clickjacking is. You have logged into your banking application. You've completed your business there, but didn't bother to log out. Then you go to your favorite hacker's forum, where not all individuals are always trustworthy.

You stumble upon a page that a hacker was putting there, and it has all these form fields in it. It doesn't seem related to anything. You just click your way through this page, but actually this page is used to cover the fund transfer form from your banking application.

You know, in some sense, you could actually construct a clickjacking attack in a manner that would even, in some cases, have you actually surrender maybe your login credentials, and then go from login to fund transfer or any other activity, all through pages that look totally different and seem to be totally unrelated to the actual application being attacked.

Brian Contos: I see. So it's actually taking advantage of a couple things. One, the fact that I have some credentials that are active because I haven't logged out. And two, the fact that this compromised site can use some type of overlay that's transparent to me. I simply don't see it. So it's taking advantage of those two pieces in tandem.

Amichai Shulman: The second one is actually the site being able to be displayed through HTML frames.

Brian Contos: Let me ask you, something like this just seems like it's probably caught a lot of users, and probably a lot of application administrators, off-guard. Have there been some examples of this in the media?

Amichai Shulman: Actually, yes. I think that the most notorious example so far is the clickjacking attack mounted against the Twitter social networking site. In fact, there were recently two individual incidents in which massive amounts of Twitter users fell victim to a clickjacking scheme through, I think it was, an advertisement page. We are seeing real usage of this technique in the wild.

Brian Contos: Now maybe Twitter's not a good example of this, but do these attacks seem to be financially motivated at this point? What seems to be the main driver behind them?

Amichai Shulman: This incident wasn't what we'd say a "monetized" type of incident, but definitely the incentive here would be financial. There's no doubt about it. For example, one of the things that I can do with clickjacking is probably change passwords for users in the application that I am hijacking. If I know what the new credentials are, because they were typed into a presumably non-related field in a different page, then I can later go on and log into the application on behalf of the victim, do whatever I have to do or want to do, transfer funds, get money, buy stuff on behalf of the victim, and then monetize the attack.

I don't think, today, that you can point to any technique that is not eventually used for financially motivated attacks.

Brian Contos: So who's at risk, really, for this? Is it a particular type of programming language that people are using on their application? Is it a particular type of server? What are the risky elements of this?

Amichai Shulman: I think it's a particular type of user. I think the attack is technology agnostic. You could actually find vulnerable applications of every technology on every type

Clickjacking with Amichai Shulman

of server today. I think that most applications would be, in fact, vulnerable to clickjacking today. I think that it's a certain usage pattern of users that promotes this type of attack, because you would need to be a user that uses a multi-tab browser, and that uses the same browser window for sensitive applications and non-sensitive applications. You would be the user that does not log out from sensitive applications once you're done with them.

I know that I'm describing most users today, but it's essentially a usage type of problem, rather than a specific technology that is vulnerable to this attack.

Brian Contos: So besides signing out every time you leave, let's say, your online banking site, what can be done to mitigate these threats? Maybe both from a user perspective, and maybe an organizational perspective?

Amichai Shulman: From a user perspective, [as] you said, bother to log out from sensitive applications once you're through with them. Don't use the same browser window for your banking application and general surfing. Those are always generally good guidelines. They would protect you against a lot of the browser-related vulnerabilities as well. But it doesn't take the responsibility off from application owners to have their application immunized against this type of attack.

Mitigating is not very easy. While this type of attack looks a little bit like cross-site request forgery, the measure that is usually used against cross-site request forgery attack, which is embedding a random nonce or a random challenge into each and every form in the application, would not help in that case.

Because when the attack is launched, from the server perspective, the interaction is identical to the interaction with a normal user, including all random challenges and everything.

One way to mitigate a lot of these attacks is to take care, in the application level, to look at the referrer field coming from the browser, because that would actually make a lot of difference between whether this is part of a normal application flow, or an interrupt coming into the application flow from actually a different page.

So this is one thing that could mitigate it, although there are some issues with referrer-based protection. Mainly, there's an issue of many users bothering to mask their referrer field for some privacy-related issues.

Other alternatives include adding frame-busting code to each of your application pages, or at least the sensitive ones. Frame-busting code is a piece of JavaScript code inserted into the page making sure that the page would never be inserted within a frame, which is a technique used by clickjacking.

It's not always possible, in particular, if you have an application that does use frames. If you do use frames in your application, you could do the reverse of the random challenge, which means you put a random challenge into the outer frame and make sure that each inner frame looks for that challenge in the outer frame. It ensures that the application forms are always used within the context of the application frame, rather than an external frame.

So those are quite a few changes that you need to do in your application code. The only method that does not require immediate changes to application code is the referrer-based method, which can actually be employed using a web application firewall.

Clickjacking with Amichai Shulman

Brian Contos: Well, Amichai, obviously this is a relatively complex topic, especially when not looking at a visual representation of clickjacking. Where can our listeners go to find out a little bit more about what clickjacking is?

Amichai Shulman: I think that, first and foremost, we must give praise to Jeremiah Grossman from WhiteHat Security, who was the one to actually put a name on this type of attack and beautifully described it in a whitepaper and a presentation. You can go out and find his initial presentation. We in Imperva gave a webinar not long ago in which we have demonstrated this attack. You can then beautifully see the fake page as you would see it in a real attack, and then there's a button that actually makes it transparent, allowing you to get the clear view of what's really behind it and get the real meaning of clickjacking.

So, by all means, go to our site. Look for the webinar, and you'll see a beautiful example there.

Brian Contos: Great. Well, thank you so much, Amichai. As always, it's a pleasure having you on the call.

Amichai Shulman: Thank you, Brian. It was my pleasure to be here.

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

