Disclosures, Jenkins, Conferences, and the Joys of 0Days

Posted on **November 17,2016** by **lynx**

DeepSec 2016 was great. We have slightly recovered and deal with the aftermath in terms of administrivia. As announced on Twitter, we would like to publish a few thoughts on the remote code execution issue found by Matthias Kaiser. He mentioned the possibility in this presentation titled *Java Deserialization Vulnerabilities - The Forgotten Bug Class*. First let's explain some things about how DeepSec runs the Call for Papers, the submissions, and the conference.

During the Call for Papers process our speakers send us title, abstract, and mostly an in-depth description of the presentation's content. This means that we usually know what's going to happen, except for the things that are actually said and shown during the presentation slot. Since we do not offer any live video streams and publish all presentation slides after we have given the speaker a chance to redact and change things, the disclosure of anything is limited to the audience at DeepSec. Even if you do a full disclosure, it is technically not. That's a fact, not an excuse. Furthermore we support all of our speakers during the submission process, on stage, and after that. If you as a conference do not give this support for anything that might happen, then what's the point in inviting someone? Once you publish the schedule, you're in. Don't cop out!

Any kind of disclosure comes with a discussion on how to do it. Basically you get disclosure in all shades and all flavours. In our heart we very much like full disclosure (we grew up with reading Bugtraq and stuff like that years before). We suggest reading the superb article Full Disclosure is a necessary evil written by Aleph One in 2001. True, most vendors/developers/communities like to get some advance information on critical bugs in order to fix the problem. Sadly this statement is based on the assumption that whoever produces the code is willing to do this. There are records of bugs that were critical and weren't fixed for years (or longer). This is still the case. So, no matter what flavour of disclosure you like, the information about the bug has to be published, and it has to be published with a fixed deadline. There is nothing to postpone. The bug is real, it affects users, it is being exploited. Time is running out. End of story.

Lastly we don't like the term 0day (or zero-day). It's a fancy word. It sounds dangerous (it might be, it might be not, most of the time it is, but then it depends on what the affected code does). The 0day is at the end of the life cycle from vulnerability to bug and to a working and tested exploit. In order to wreak havoc you have to do some software development including testing to come up with results that can be recreated. Neither a vulnerability nor a bug is a 0day.

Of course we understand vendors, developers, and communities that care about the security record of their code. Plus we like security researchers who don't give out the details too early. Thanks to Matthias Kaiser for being responsible and professional. In case you got the wrong impression from a medium that asks you to explain quantum field theory in 140 characters, then something went wrong. We are sorry for any conclusions derived from instant news messages.

We hope to see vendors, developers, security researchers, user, and everyone else at DeepSec 2017. We can discuss the joys of 0days live with as many characters and coffee as we want.

Posted in:Conference,Discussion,High Entropy | Tagged:Bug,DeepSec,Exploit,Software | With 4 comments