The Honeynet Project

Cuckoo Sandbox meets Mac OS X

Tue, 11/10/2015 - 20:54 — jurriaan.bremer

Twitter Facebook LinkedIn

Posting this blogpost on behalf of Dmitry Rodionov.

Hi there! I'm Dmitry Rodionov and this summer I've been working on an OS X analyzer for Cuckoo Sandbox project.

Cuckoo Sandbox

First things first: what is Cuckoo Sandbox? Imagine a box you can put any suspicious program or script into and immediately receive a complete description of what this program is and what it does. Well, that's Cuckoo!

Cuckoo launches every program in a separate virtual machine (a sandbox), so there is no risk for your own computer to be infected with a virus or to leak private information.

As you may guess already, Cuckoo Sandbox is a complex software that consists of many different modules. And one of the vital parts of this system is an analyzer. An analyzer is a program that runs inside a virtual machine (yes, right next to the malware -- just like a spy!) and collects every bit of information about a target and its behaviour: what kind of APIs did it call (arguments and return value included!), what files did it touch, a summary of network activity and many more including even screenshots ヽ(^o^)ノ

Obviously, this analyser must know enough about its environment (just like a real spy must know a lot about his enemies), so Cuckoo Sandbox has different analysers for different operation systems: one for Windows, one for GNU\Linux and now one for OS X ヽ(ﾟДﾟ)ﾉ

OS X analyzer implementation details and other interesting stuff

So let's talk about this analyzer module I've been building for three summer months. There aren't a lot of for one application to see what another one (let's call it a target) is doing:

1) it may inject a piece of code into this target program; this injected piece of code will live inside the app just like the original one and will have the same capabilities and rights. OK, but how can this help us? The thing is: you can override any function within the target application with our own one and thus react on every function call.

2) The second way is to use some third-party tool for dynamic tracing: like strace or, on OS X, DTrace. This tools allows you to be notified when something happens in a target process. So basically, they do all the dark magic for us and we only have to handle results.

Despite the fact that Cuckoo Sandbox analyzer for Windows utilises the first option, I chose not to perform code-injection but to take advantage of DTrace because it's a built-in utility on OS X and why not?

Aaaaand then I spent the entire summer fighting with DTrace. Sure it's a super-powerful system, but on the other hand it has so many limitations (e.g., you can't use any loops or if/else-statements in there) and tricky gotchas (e.g., how would you follow children of a target process?) In other words, one has to pay a lot of attention to details — otherwise the whole thing simply won't work.

Here're a few problems that I've encountered during my DTrace crash-course:

1) My DTrace scripts must output information in such a way that other scripts (in Python) could parse it. Well, the choice of format wasn't a problem: JSON is simple and even human-readable, but what I had to figure out was how to compose a single JSON entry from data that comes from different DTrace probes. I ended up with a simple stack-like data structure for function arguments and return values -- and it worked out pretty well!

2) Sometimes our targets spawn one or more children processes that may also do some interesting stuff. If this is the case, we must also monitor their activities by attaching a new instance of DTrace to them. It sounds pretty easy yet there's no way to do this in DTrace out of box, so I had to implement this kind of monitoring and DTrace re-attaching myself. That was really challenging and fun (~￣︶￣)~

It's not that these "problems" were really hard or anything, but I've never been working with DTrace before and thus it took me a while to figure everything out. I'm going to write a few blog posts about dark corners of DTrace on my blog, so check it out if you're interested!.

When I wasn't busy fighting with DTrace, I was trying to establish a communication channel between the analyzer and Cuckoo host (for sending analysis results, obviously), writing tests (gosh, I don't trust neither myself nor computers much!) and building base analysis packages for OS X executables (Mach-O), Cocoa application (.app), Bash scripts (.sh) and Zip archives.

Setup instructions and usage information

I have a pretty detailed setup guide for you here. Please, check it out and ask any questions if something remains unclear or simply wrong. Thanks!

Last but not least

I'd like to thank my wonderful mentors: Jurriaan Bremer and Alessandro Tanasi. I guess I couldn't have done all this without their invaluable support and honest feedback.
Love you guys! ❤\(//˚▽˚//)

Our feedback

The Cuckoo Sandbox team is very proud of Dmitry's work throughout the summer, he did a great job. I'm happy to announce that Dmitry will be continuining to work on Mac OS X analysis as part of his studies at the University, so we're expecting Mac OS X analysis to become more and more mature as he works on it :-)

jurriaan.bremer's blog Twitter Facebook LinkedIn