HaXmas: The True Meaning(s) of Metasploit

HaXmas: The True Meaning(s) of Metasploit - @todb, Tod Beardsley, Rapid7 Blog.

- Published: 2017-12-25
- Original: <https://blog.rapid7.com/2017/12/25/haxmas-the-true-meaning-s-of-metasploit/>
- Preserved from: https://blog.rapid7.com/2017/12/25/haxmas-the-true-meaning-s-of-metasploit/ (stored) on 2026-08-09
- Capture timestamp: 20191224195838
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

Happy HaXmas, everyone! 'Tis the season for storytelling and reminiscing about the year gone by, so I wanted to take a moment to highlight one particular Metasploit module that landed in my lap this year: the storied SOP Bypass Module that one Mishra @RootUp Dhiraj proposed for the Metasploit Framework back in September. This module is what taught me the True Meaning(s) of Metasploit: It's a universal language for exploit development, a grab bag of a million useful hackery tricks, and an irresistible platform for attracting rarified exploitation talent. Okay, it's three true meanings. A HaXmas miracle.

This story begins with a note from Brent @busterbcook Cook flagging this issue opened against the Metasploit Framework that seemed to include an unpatched, zero-day vulnerability in the Samsung Internet Browser, a browser that's scoring in the 100,000,000 - 500,000,000 tier of Android downloads. That sounded like it might be a big deal, because people rarely disclose 0day in the Metasploit pull request queue, and the Samsung Browser is a pretty popular not-Chrome Android browser.

Being fans of coordinated disclosure, we checked with Samsung, and indeed, this issue was reported, and a patch was scheduled to be released. However, due to some language barriers, it ended up being pretty difficult to tell what, exactly, was being exploited in RootUp's proof-of-concept, so I asked him to close out the issue and put together a proper Metasploit module.

And boy howdy, am I glad I did. Rather than trying—and failing—to decode what this exploit's intentions were from the original, kind of slap-dash Javascript snippit, RootUp rose the the challenge, and put together a Metasploit module in pretty short order. This brings me to what I feel like is one of Metasploit's strong suits: providing a *lingua franca* for expressing exploits and vulnerabilities. Through Metasploit's conventions and structure, it gets pretty easy to cut to the

chase of an exploit, even when there are language and cultural barriers between researchers. It's really pretty magical.

So, with this first version in hand, I was prepared to offer a more formal review of the module, and hopefully tease out exactly what the vulnerability being exploited is. To my frank surprise, RootUp remained enthusiastic after my initial comments, which were essentially all, "wait what?" Again, I think that working in Metasploit Framework helped to smooth over the communication barriers between us, and gave us a roadmap to exploitation for this browser.

Turns out, the crux of the issue was this: When the Samsung Internet browser opens a new tab in a given domain (say, google.com) through a Javascript action, that Javascript can come in after the fact and rewrite the contents of that page with whatever it wants. This is a no-no in browser design, since it means that Javascript can violate the Same-Origin Policy, and can direct Javascript actions from one site (controlled by the attacker) to act in the context of another site (the one the attacker is interested in). Essentially, the attacker can insert custom Javascript into any domain, provided the victim user visits the attacker-controlled web page first.

Now that we both had the same understanding of the bug, the matter of writing the exploit was pretty straight forward. To be honest, I'm not much of a browser exploit kind of guy—I deal more in goofy proprietary protocols and easy, old-timey memory management vulns—but I cribbed from the work of Rafay Baloch and Joe Vennix, two Metasploit browser exploiters of HaXmas Past. This is an example of the second most valuable feature of Metasploit: cargo-culting from other Metasploit modules is often the fastest way to get things rolling. You really don't have to be a domain expert in the target technology to get a solid exploit together.

Lucky for both of us, Brendan Coles and Jeffery Martin piled on with more review notes, and after some back and forth across time zones, we ended up landing a pretty decent Metasploit module just a few days ago. This highlights the third most magical feature of Metasploit: Do something kind of neat, and incredibly helpful, knowledgeable, and talented people pop out of the Github woodwork to help you along. It really doesn't take much more than that.

With that, behold, the Samsung Internet Browser SOP Bypass module, where, as long as you can trick your victim into visiting your Metasploit-served webpage, you have a pretty decent chance of tricking them out of a username or password—or, if you want to do something else, like snag a session cookie or hook the whole browser session, just set a `CUSTOM_JS` advanced option to fire that off.

And, as a bonus, not only did RootUp stick through all the review and all the updates like a true champion, he closed things out with a couple quick screencasts of the module in action. Below is a screencast of the effect on the client side (which is the exciting bit), and you can hop over here to see what's going on on the Metasploit console side.

Happy HaXmas!
