- **Regan Robertson** Mod • 14 days ago

  Good Afternoon,

  The video will start on this page at 12:00pm, and you may have to hit play or unmute. As a reminder this site works best in a Chrome Browser. You can write in comments through this feature to continue the conversation or ask questions. Please login or sign up for a Disqus account to participate in the discussion boards.

  - Reply
  - Share ›

- 

  Matt Mickelson • 14 days ago

  Thank you all for attending the talk... here's to a healthy discussion.

  - Reply
  - Share ›

- 

  Fabrizio Bertocci • 14 days ago • edited

  Now it's ~30k...

  ```
  ls -l /bin/true
  -rwxr-xr-x 1 root root 30904 Jan 18 2018 /bin/true
  ```

  (Ubuntu 20.xx)

  - 1
  - Reply
  - Share ›

Nathaniel Husted Fabrizio Bertocci • 14 days ago • edited

Hopefully that's like ~30K of security built in. ;-)

Reply

Share ›

Jason H Li • 14 days ago

Less is more and better. But how do you know the job is done right? Well, the original bloated SW was not right anyways. But now as a 'medical doctor' performing surgery, the first is to do no harm. Any thoughts?

Reply

Share ›

Matt Mickelson Jason H Li • 14 days ago

Indeed. The "bottom-up formal method" can help prove the do-no-harm part... and then go further with more properties.

Reply

Share ›

Renato Levy • 14 days ago • edited

The problem on removing "dead" functionality, is exactly on knowing your use cases well enough that you don't run on the need for it at runtime....Specially when you do it at byte level...

Reply

Share ›

Matt Mickelson Renato Levy • 14 days ago

We make a fine distinction between debloating (removal of dead/unreachable code) and feature removal. Dead code shouldn't be run by your users... but could be accessed by an attacker. Moreover, in any case you have the original executable, so you could always rerun any transformations later if you find you need "that thing you removed" before.

Reply

Share ›

Renato Levy Matt Mickelson • 14 days ago

the concept of overlay can also be placed here. You can have a scratch pad area that you bring the code back for a short period of time. If you have a control flow analysis based on dominant node, you may be able to know enough in advance if that function is needed, to avoid performance impacts.

- Reply
- Share ›

Ryan Craven Renato Levy • 14 days ago

Doing this you must be careful though that in the event an attacker triggers the functionality that it's *not* brought back. There's also the case where a well-intentioned user is at odds with a sysadmin. I'm not sure how to safely make the distinction of whether an action you had debloated is ok and you were just too aggressive. All depends on the deployment environment & context.

- Reply
- Share ›

Aleksey Nogin • 14 days ago

You make the claim that transforms you make to the binary sever the connection to the original, requiring you to verify the transformed binary. Would not you get more "mileage" out of verifying the original and proving the transformer correctness?

Reply

Share ›

Matt Mickelson Aleksey Nogin • 14 days ago

I'd suggest you get the most mileage out of verifying the actual executable (as that's still the thing that runs on your system)... anything else has the compiler in the chain.

Reply

Share ›

Ihor Kuz Matt Mickelson • 14 days ago

This is a topic that comes up regularly when we do binary verification: is it better to do do the binary verification or to verify the compiler (bottom up vs top down)? What's the right answer?

Reply

Share ›

Stuart Card • 14 days ago

How & how soon do we think bottom-up methods of verification can be sufficiently automated to enable dynamic remote attestation of properties?

- Reply
- Share ›

Matt Mickelson Stuart Card • 14 days ago

Fantastic question... the sooner the better in my opinion. This is the group of research teams that is probably leading the way.

- Reply
- Share ›

Jason H Li • 14 days ago

Another question - while 'dynamic attack surface' is cool, it is critical to manage dynamics within platform and mission context to ensure mission resilience or even basic operation. Moving by itself could hurt us too. Some hard lessons learned but worth it.

- Reply
- Share ›

Matt Mickelson Jason H Li • 14 days ago

Indeed... you have to couple any transformations with some operational activities to manage the different variants

Reply

Share ›

Fabrizio Bertocci • 14 days ago • edited

Are there any existing (proof of concept) examples that show this solution will actually work (given enough time and resources)? I guess the critical component is the decompiler to transform from machine code to a higher language.

Reply

Share ›

Matt Mickelson Fabrizio Bertocci • 14 days ago

There are a number of tools from the TPCP program that have already established POCs... diversification tools, debloating tools, and verification tools. Early results on debloating are anywhere between 30%-60% reduction of attack surface.

Reply

Share ›

Nathaniel Husted Matt Mickelson • 14 days ago

The Virtual Summer School sponsored by ONR and organized by Purdue University this year was a great environment to learn and play around with some of the more mature TPCP tools.

Reply

Share ›

Matt Mickelson Nathaniel Husted • 14 days ago

I was going to plug SSSS'20 as well... you beat me to it... thanks :)

1

Reply

Share ›

Olin Sibert • 14 days ago

The digital rights management world relies heavily on automated program diversity and also includes transformations designed to obfuscate the high-level algorithms, not just to be different. Transforming a chunk of somewhat comprehensible binary code into an equivalent FSM is a strong deterrent to reverse engineering.

- Reply
- Share ›

Olin Sibert Olin Sibert • 14 days ago

There has been a big debate in those circles about whether diversity (or obfuscation) is more effective if achieved by automated rewriting of source code or binary code. Rewriting source code can yield dramatically different results post-optimization, but rewriting binary code is a lot simpler and it's easier to be confident of results.

- Reply
- Share ›

Jason H Li Olin Sibert • 14 days ago

Indeed. I've seen many research works arguing either way.

- Reply
- Share ›

Matt Mickelson Olin Sibert • 14 days ago

At ONR, there's a nontrivial focus on legacy code... so the abilty to transform without source code is huge.

- Reply
- Share ›

Matt Mickelson Olin Sibert • 14 days ago

Indeed. All cyber is dual use... there are certainly use cases focused on defending and obfuscating

- Reply
- Share ›

Renato Levy • 14 days ago

As you lift the code, there are also ways to modify the binary, while keeping tabs and modify the source accordingly. Off course, for the part where the source is available.

Reply

Share ›

Regan Robertson Mod • 14 days ago

We hope everyone is enjoying the second day of the summit. We are going into the lunch break and will be back at 1pm!

Reply

Share ›

Jacob Saina • 14 days ago

@**Regan Robertson** will the slide decks and/or videos be made available for after the summit?

Reply

Share ›

—

Regan Robertson Mod Jacob Saina • 14 days ago

Hi Jacob, The videos will stay on this hub for the next 90 days. We do not have the slide decks, but you can reach out to the particular presenter to ask for a copy.

- Reply
- Share ›