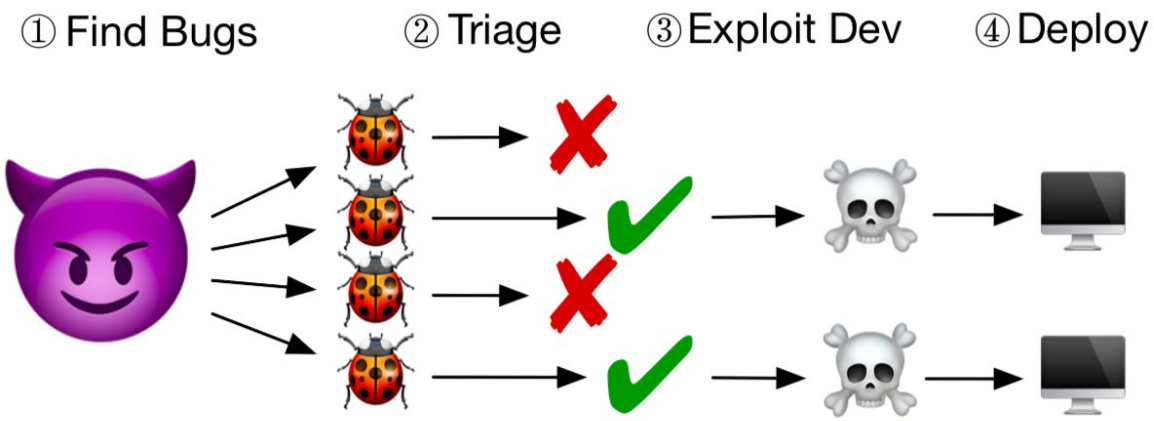
A new defensive technique could hold off attackers by making software buggier

August 10 2018, by Ingrid Fadelli



Simplified attacker workflow. Attackers find bugs, triage them to determine exploitability, then develop exploits and deploy them to their targets. Credits: Hu, Hu & Dolan-Gavitt.

Researchers at New York University have recently devised a new cyber defense technique, which works by adding so-called "chaff bugs," non-exploitable bugs, rather than eliminating existing ones. A pre-print version of their inventive study was uploaded to *ArXiv* last week.

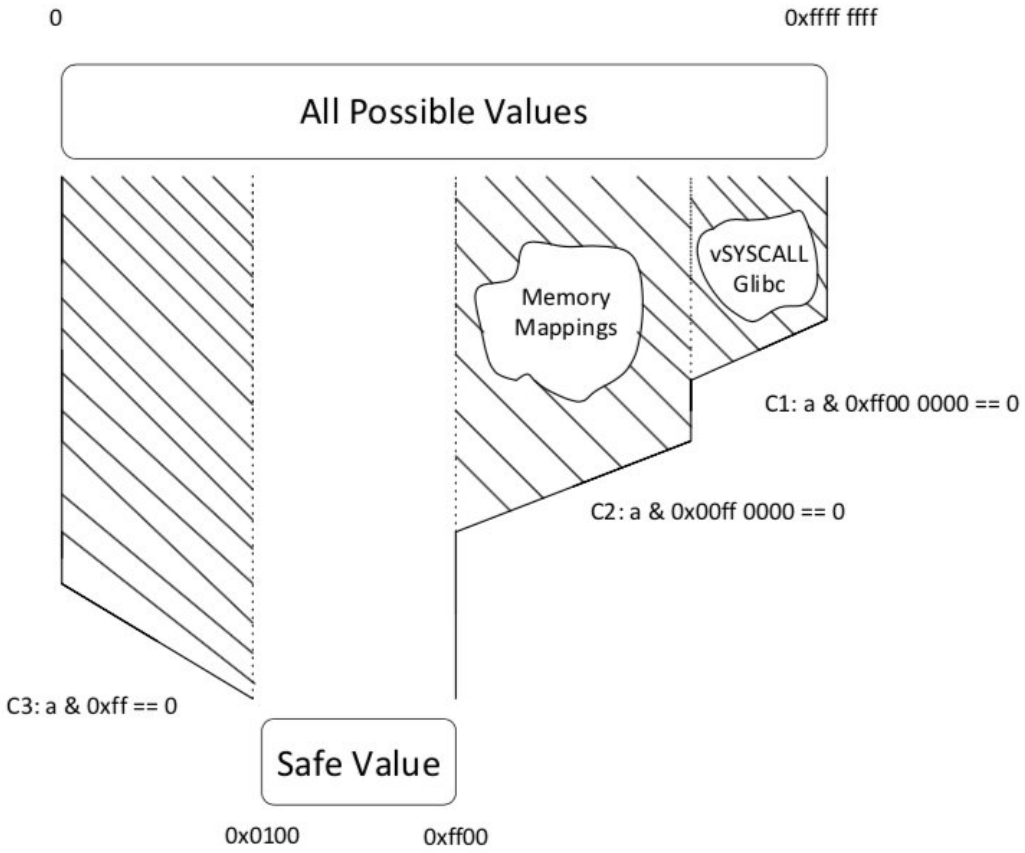
Every day, increasingly sophisticated attackers find bugs in computer software, evaluate their exploitability, and develop ways of exploiting

them. Most existing defense techniques are aimed at eliminating these bugs, limiting them, or adding mitigations that can make exploitation more challenging.

"Our project was based on some earlier work we did in collaboration with MIT Lincoln Lab and Northeastern University on evaluating different strategies for finding bugs in software," Brendan Dolan-Gavitt, one of the researchers who carried out the study, told Tech Xplore. "To do that, we built a system that could put thousands of bugs into a program, so that we could then measure how effective each bug-finding strategy was at detecting our bugs."

After they developed this system, the researchers started considering its possible applications in the context of improving software security. They soon realized that there is a bottleneck in how attackers typically find and exploit bugs in software.

"It takes a lot of time and expertise to figure out which bugs are exploitable and develop a working exploit," Dolan-Gavitt explained. "So we realized that if we could somehow make sure that all the bugs we added were non-exploitable, we could overwhelm attackers, drowning them in a sea of enticing-looking but ultimately harmless bugs."



An overconstrained value bug. By adding constraints along the path leading to the bug, we gradually eliminate unsafe values. Credits: Hu, Hu & Dolan-Gavitt.

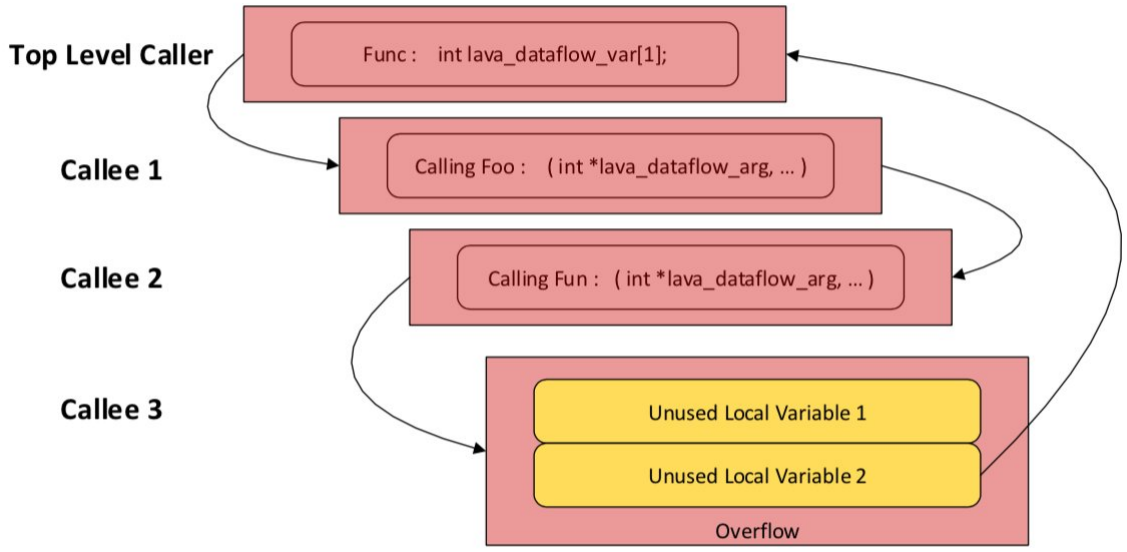
Adding huge numbers of bugs that are provably—but not obviously—non-exploitable could confuse attackers, driving them to waste their time and efforts on seeking exploits for these intentionally placed bugs. The researchers called this new method of deterring attackers "chaff bugs."

"Our system, which automatically adds chaff bugs, is intended to be used when developers are building software," Dolan-Gavitt said. "We use two

strategies to ensure that the bugs are safe: either by guaranteeing that the attacker can only corrupt data that's not being used by the program, or by making sure that the values the attacker can inject are restricted to ranges that we can determine are safe."

In their recent study, the researchers used these two strategies to add thousands of non-exploitable bugs to real-world software automatically, including web server NGINX and encoder/decoder library libFLAC. They found that the functionality of the software was unharmed, and that the chaff bugs appeared exploitable to current triage tools.

"One of the most interesting findings is that it's possible to construct these non-exploitable bugs automatically, in a way that we can tell they're not exploitable, but it's hard for an attacker to do the same," Dolan-Gavitt said. "It wasn't obvious to us when we started that this would be doable. We also think this approach of applying a sort of economic logic – finding out what attacker resources are scarce and then trying to target them – is a fruitful area to explore in software security."



An unused variable bug. Data flow is added to propagate the “unused” values outside the initial scope, hiding the fact that they are actually unused. Credits: Hu, Hu & Dolan-Gavitt.

While their study gathered promising results, several challenges still need to be overcome for this approach to become practically viable. Most importantly, the researchers need to figure out a way of making these non-exploitable bugs entirely indistinguishable from real bugs.

"Right now, the bugs we add are pretty artificial-looking, so attackers could use this fact to ignore ours when looking for exploitable bugs," Dolan-Gavitt said. "Our main focus at the moment is on finding ways to make the bugs we've added look more like naturally occurring bugs, and then running experiments to show that attackers really are fooled by our chaff bugs."

More information: Chaff Bugs: Deterring Attackers by Making Software Buggier, arXiv: 1808.00659v1 [cs.CR]. arxiv.org/abs/1808.00659

Abstract
Sophisticated attackers find bugs in software, evaluate their exploitability, and then create and launch exploits for bugs found to be exploitable. Most efforts to secure software attempt either to eliminate bugs or to add mitigations that make exploitation more difficult. In this paper, we introduce a new defensive technique called chaff bugs, which instead target the bug discovery and exploit creation stages of this process. Rather than eliminating bugs, we instead add large numbers of bugs that are provably (but not obviously) non-exploitable. Attackers who attempt to find and exploit bugs in software will, with high probability, find an intentionally placed non-exploitable bug and waste

precious resources in trying to build a working exploit. We develop two strategies for ensuring non-exploitability and use them to automatically add thousands of non-exploitable bugs to real-world software such as nginx and libFLAC; we show that the functionality of the software is not harmed and demonstrate that our bugs look exploitable to current triage tools. We believe that chaff bugs can serve as an effective deterrent against both human attackers and automated Cyber Reasoning Systems (CRSes).