lcamtuf's blog

August 31, 2015

Understanding the process of finding serious vulns

Our industry tends to glamorize vulnerability research, with a growing number of bug reports accompanied by flashy conference presentations, media kits, and exclusive interviews. But for all that grandeur, the public understands relatively little about the effort that goes into identifying and troubleshooting the hundreds of serious vulnerabilities that crop up every year in the software we all depend on. It certainly does not help that many of the commercial security testing products are promoted with truly bombastic claims - and that some of the most vocal security researchers enjoy the image of savant hackers, seldom talking about the processes and toolkits they depend on to get stuff done.

I figured it may make sense to change this. Several weeks ago, I started trawling through the list of public CVE assignments, and then manually compiling a list of genuine, high-impact flaws in commonly used software. I tried to follow three basic principles:

- For pragmatic reasons, I focused on problems where the nature of the vulnerability and the identity of the researcher is easy to ascertain. For this reason, I ended up rejecting entries such as CVE-2015-2132 or CVE-2015-3799.

- I focused on widespread software - e.g., browsers, operating systems, network services - skipping many categories of niche enterprise products, Wordpress add-ons, and so on. Good examples of rejected entries in this category include CVE-2015-5406 and CVE-2015-5681.

- I skipped issues that appeared to be low impact, or where the credibility of the report seemed unclear. One example of a rejected submission is CVE-2015-4173.

To ensure that the data isn't skewed toward more vulnerable software, I tried to focus on research efforts, rather than on individual bugs; where a single reporter was credited for multiple closely related vulnerabilities in the same product within a narrow timeframe, I would use only one sample from the entire series of bugs.

For the qualifying CVE entries, I started sending out anonymous surveys to the researchers who reported the underlying issues. The surveys open with a discussion of the basic method employed to find the bug:

```
  How did you find this issue?

  ( ) Manual bug hunting
  ( ) Automated vulnerability discovery
  ( ) Lucky accident while doing unrelated work
```

If "manual bug hunting" is selected, several additional options appear:

```
  ( ) I was reviewing the source code to check for flaws.
  ( ) I studied the binary using a disassembler, decompil
er, or a tracing tool.
  ( ) I was doing black-box experimentation to see how th
e program behaves.
  ( ) I simply noticed that this bug is being exploited i
n the wild.
  ( ) I did something else: _____________________
```

Selecting "automated discovery" results in a different set of choices:

```
  ( ) I used a fuzzer.
  ( ) I ran a simple vulnerability scanner (e.g., Nessus)
.
  ( ) I used a source code analyzer (static analysis).
  ( ) I relied on symbolic or concolic execution.
  ( ) I did something else: _____________________
```

Researchers who relied on automated tools are also asked about the origins of the tool and the computing resources used:

```
  Name of tool used (optional): _____________________

  Where does this tool come from?

  ( ) I created it just for this project.
  ( ) It's an existing but non-public utility.
  ( ) It's a publicly available framework.

  At what scale did you perform the experiments?
```

```
  ( ) I used 16 CPU cores or less.
  ( ) I employed more than 16 cores.
```

Regardless of the underlying method, the survey also asks every participant about the use of memory diagnostic tools:

```
  Did you use any additional, automatic error-catching to
ols - like ASAN
  or Valgrind - to investigate this issue?

  ( ) Yes. ( ) Nope!
```

...and about the lengths to which the reporter went to demonstrate the bug:

```
  How far did you go to demonstrate the impact of the iss
ue?

  ( ) I just pointed out the problematic code or function
ality.
  ( ) I submitted a basic proof-of-concept (say, a crashi
ng test case).
  ( ) I created a fully-fledged, working exploit.
```

It also touches on the communications with the vendor:

```
  Did you coordinate the disclosure with the vendor of th
e affected
  software?

  ( ) Yes. ( ) No.

  How long have you waited before having the issue disclo
sed to the
  public?

  ( ) I disclosed right away. ( ) Less than a week. ( ) 1
-4 weeks.
  ( ) 1-3 months. ( ) 4-6 months. ( ) More than 6 months.

  In the end, did the vendor address the issue as quickly
 as you would
  have hoped?

  ( ) Yes. ( ) Nope.
```

...and the channel used to disclose the bug - an area where we have seen some stark changes over the past five years:

```
  How did you disclose it? Select all options that apply:

  [ ] I made a blog post about the bug.
  [ ] I posted to a security mailing list (e.g., BUGTRAQ)
.
```

```
  [ ] I shared the finding on a web-based discussion foru
m.
  [ ] I announced it at a security conference.
  [ ] I shared it on Twitter or other social media.
  [ ] We made a press kit or reached out to a journalist.
  [ ] Vendor released an advisory.
```

The survey ends with a question about the motivation and the overall amount of effort that went into this work:

```
  What motivated you to look for this bug?

  ( ) It's just a hobby project.
  ( ) I received a scientific grant.
  ( ) I wanted to participate in a bounty program.
  ( ) I was doing contract work.
  ( ) It's a part of my full-time job.

  How much effort did you end up putting into this projec
t?

  ( ) Just a couple of hours.
  ( ) Several days.
  ( ) Several weeks or more.
```

So far, the response rate for the survey is approximately 80%; because I only started in August, I currently don't have enough answers to draw particularly detailed conclusions from the data set - this should change over the next couple of months. Still, I'm already seeing several well-defined if preliminary trends:

- The use of fuzzers is ubiquitous (incidentally, of named projects, *afl-fuzz* leads the fray so far); the use of other automated tools, such as static analysis frameworks or concolic execution, appears to be unheard of - despite the undivided attention that such methods receive in academic settings.

- Memory diagnostic tools, such as ASAN and Valgrind, are extremely popular - and are an untold success story of vulnerability research.

- Most of public vulnerability research appears to be done by people who work on it full-time, employed by vendors; hobby work and bug bounties follow closely.

- Only a small minority of serious vulnerabilities appear to be disclosed anywhere outside a vendor advisory, making it extremely dangerous to rely on press coverage (or any other casual source) for evaluating personal risk.

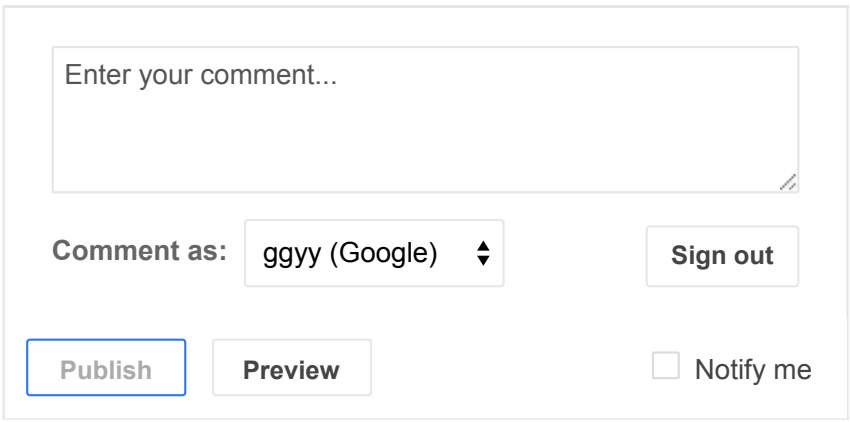
Of course, some security work happens out of public view; for

example, some enterprises have well-established and meaningful security assurance programs that likely prevent hundreds of security bugs from ever shipping in the reviewed code. Since it is difficult to collect comprehensive and unbiased data about such programs, there is always some speculation involved when discussing the similarities and differences between this work and public security research.

Well, that's it! Watch this space for updates - and let me know if there's anything you'd change or add to the questionnaire.

No comments:

Post a Comment



Home Older Post

Subscribe to: Post Comments (Atom)