THE CYBER SECURITY PROJECT

Taking Stock

Estimating Vulnerability Rediscovery

Trey Herr

Bruce Schneier

Christopher Morris

HARVARD Kennedy School
BELFER CENTER
for Science and International Affairs

PAPER
ORIGINAL: JULY 2017
REVISED: OCTOBER 2017

The Cyber Security Project
Belfer Center for Science and International Affairs
Harvard Kennedy School
79 JFK Street
Cambridge, MA 02138

www.belfercenter.org/Cyber

Design & Layout by Andrew Facini

Cover image and opposite page 1: A screenshot from the Google Chrome source code.

Printed in the United States of America

THE CYBER SECURITY PROJECT

Taking Stock

Estimating Vulnerability Rediscovery

Trey Herr

Bruce Schneier

Christopher Morris

HARVARD Kennedy School
BELFER CENTER
for Science and International Affairs

PAPER
ORIGINAL: JULY 2017
REVISED: OCTOBER 2017

About the Authors

Trey Herr, Post-Doctoral Fellow
Belfer Center Cyber Security Project, Harvard Kennedy School
trey_herr@hks.harvard.edu

Bruce Schneier, Research Fellow and Lecturer
Belfer Center Cyber Security Project, Harvard Kennedy School
schneier@schneier.com

Christopher Morris, Research Assistant
Harvard School of Engineering and Applied Sciences
christophermorris@college.harvard.edu

Acknowledgments

This paper has been revised from its original version and the authors would like to thank several reviewers for their feedback including Nicholas Weaver and Tod Beardsley. The paper acknowledges support from the Belfer Family and the Flora and William Hewlett Foundation. This dataset and the resulting paper would not have been possible without Eduardo Vela Nava and Andrew Whalley of Google; Casey Ellis and Payton O'Neal of Bugcrowd; Rich Salz of OpenSSL; Richard Barnes, Dan Veditz, and Al Billings of Mozilla; and Art Manion and Allen Householder of CERT/CC. Special thanks are also owed to Martin Shelton, Katherine Bjelde, and Jim Waldo for their help cleaning and formatting the data. The authors would additionally like to thank Beth Friedman, Annie Boustead, Sasha Romanosky, Jay Healey, Mailyn Fidler, Thomas Dullien, Herb Lin, Fabio Massacci, Gary Belvin, Beau Woods, Tudor Dumitras, Ben Laurie, and the Belfer Cyber Security team for their feedback.

Table of Contents

```
#endif  // #if defined(OS_LINUX)

// This code is specific to the Windows-only ProfileImportExport field trial.
void AddProfileImportingTime(const char* name, base::TimeDelta time) {
  const base::TimeDelta kMin = base::TimeDelta::FromMilliseconds(1);
  const base::TimeDelta kMax = base::TimeDelta::FromHours(1);
  static const size_t kBuckets = 50;

  // FactoryTimeGet will always return a pointer to the same histogram object,
  // keyed on its name. There's no need for us to store it explicitly anywhere.
  base::Histogram* counter = base::Histogram::FactoryTimeGet(
      name, kMin, kMax, kBuckets, base::Histogram::kUmaTargetedHistogramFlag);

  counter->AddTime(time);
}

}  // namespace

namespace chrome_browser {
// This error message is not localized because we failed to load the
// localization data files.
const char kMissingLocaleDataTitle[] = "Missing File Error";
const char kMissingLocaleDataMessage[] =
    "Unable to find locale data files. Please reinstall.";
}  // namespace chrome_browser

// BrowserMainParts ------------------------------------------------------------

ChromeBrowserMainParts::ChromeBrowserMainParts(
    const content::MainFunctionParams& parameters)
    : parameters_(parameters),
      parsed_command_line_(parameters.command_line),
      result_code_(content::RESULT_CODE_NORMAL_EXIT),
      startup_watcher_(new StartupTimeBomb()),
      shutdown_watcher_(new ShutdownWatcherHelper()),
      record_search_engine_(false),
      translate_manager_(NULL),
      profile_(NULL),
      run_message_loop_(true),
      notify_result_(ProcessSingleton::PROCESS_NONE),
      is_first_run_(false),
      first_run_ui_bypass_(false),
      local_state_(NULL),
      restart_last_session_(false) {
  // If we're running tests (ui_task is non-null).
  if (parameters.ui_task)
    browser_defaults::enable_help_app = false;
}

ChromeBrowserMainParts::~ChromeBrowserMainParts() {
  for (int i = static_cast<int>(chrome_extra_parts_.size()) - 1; i >= 0; --i)
    delete chrome_extra_parts_[i];
  chrome_extra_parts_.clear();
}
```

Abstract

How often do multiple, independent parties discover the same vulnerability? There are ample models of vulnerability discovery, but little academic work on this issue of *re*discovery. The immature state of this research and subsequent debate is a problem for the policy community, where the government's decision to disclose a given vulnerability hinges in part on that vulnerability's likelihood of being rediscovered and used maliciously by another party. Research into the behavior of malicious software markets and the efficacy of bug bounty programs would similarly benefit from an accurate baseline estimate for how often vulnerabilities are discovered by multiple independent parties.

This paper presents a new dataset of more than 2,600 vulnerabilities, and estimates vulnerability rediscovery across different vendors and software types. It concludes that rediscovery happens more often than the 1% to 9% range previously reported. The aggregate rediscovery rate for our dataset is 12.7%, ranging between 10.8% for Chrome between 2009 and 2017, to 21.9% for Android between 2016 and 2017. For Android and Chrome, more than 60% of all rediscovery takes place in the first month after the original vulnerability's disclosure.

These results are largely in line with those of the original version of this paper published in July 2017, and indicate that the information security community should map the impact of rediscovery on the efficacy of bug bounty programs, and policymakers should more rigorously evaluate the costs and requirements for non-disclosure of software vulnerabilities.

1. Introduction

Vulnerabilities are an important resource. Both intelligence and law enforcement activities increasingly emphasize the use of software vulnerabilities to gain access to targeted systems. These same software flaws are also a critical piece of defensive information, granting companies like Apple and open source projects like Apache insight into where there are holes in their software in need of repair. Left unfixed, software vulnerabilities provide malicious parties a point of access into any computer system running the software. Programs to pay researchers and others who disclose vulnerabilities to software developers, so-called bug bounties, are an increasingly popular way for companies to discover flaws in their code.

Underlying the choices to pay for a software vulnerability, as well as government decisions to keep some a secret, are assumptions about how often those same software flaws could be discovered by someone else, a process called rediscovery. There is very little rigorous research into how often rediscovery takes place, and yet we know it happens, sometimes in high-profile ways. For example, the Heartbleed vulnerability in OpenSSL lay dormant for three years, and yet was discovered twice within just a few days, by Neel Mehta of Google and researchers at the Finnish information security firm Codenomicon.[1] This rediscovery rate becomes particularly important if the software in question is a widely used open source project or a cryptographic library.

This paper introduces the issue of vulnerability rediscovery, presents our research, and addresses some of the larger questions raised by this phenomenon. A particularly challenging issue with rediscovery is that the rate changes over time and varies for different types of software. We collected data from multiple software vendors and an open source library to track vulnerability records and look for duplicates, where a single software flaw had been disclosed multiple times. We then used those duplicates to estimate the rate at which vulnerabilities were discovered more than once by independent parties. We also used this data to track how the rediscovery

1 Posted on April Lee, "How Codenomicon Found The Heartbleed Bug Now Plaguing The Internet," *ReadWrite*, April 13, 2014, http://readwrite.com/2014/04/13/heartbleed-security-codenomicon-discovery/.

rate can grow over time and to measure rediscovery lag—the time between when a vulnerability is first disclosed and subsequent duplicate disclosures.

The paper begins with background information on vulnerabilities and their discovery, along with a review of relevant literature and previous work. This leads into a discussion of the data collected for this project and our methodology, including the challenges present in counting vulnerabilities and measuring rediscovery. Following this are several analyses of this data, considering variation such as the vendor in question and change over time, as well as other measures such as rediscovery over time and rediscovery lag. The final two sections discuss some of the paper's limitations and then conclude by suggesting implications of this work for scholarly and policy communities.

Summary of Changes

This paper is a partially revised version of a paper previously published by the Belfer Center Cyber Security Project.[2] This paper integrates several points of feedback and includes a manually verified dataset. Changes from the original dataset have resulted in a 3% decrease in the rediscovery rates for both Chrome and Firefox, owing to newly accessible records in the bug tracking systems for both and the removal of automatically generated duplicates. This revised dataset yields total rediscovery rates of 10.8% for Chrome and 14% for Firefox. These are largely in line with our original findings and thus lead to the same conclusions. An explanation of the changes can be found below, while detailed additions have been inserted in the relevant sections throughout. In response to feedback on the original version of this paper, we have revalidated the Firefox and Chrome data, removing any records incorrectly marked as duplicates according to our coding framework, which only counts duplicates where there are two different disclosers. We have also removed a section outlining a thought experiment on intelligence community vulnerabilities.

With respect to the underlying data, we found no issues in our review of OpenSSL or Android, though we believe it to be more accurate to sample

2 Available on our GitHub. https://github.com/mase-gh/Vulnerability-Rediscovery/blob/master/Vulnerability%20Rediscovery.pdf.

only Critical, High, and Medium severity OpenSSL bugs. We did find errors in relying on duplicate labels from the Chromium database and, in expanding and revalidating the Firefox data, filtered out any records that are not from different disclosers and labeled as high or critical severity. We made a mistake in relying directly on metadata in the Chromium bug database that coded some bugs as duplicates. Many of these duplicates were automatically generated by Google's software testing infrastructure, ClusterFuzz, or were multiple reports from the same person and thus not instances of rediscovery. Reviewing the Chromium database manually, we recoded the total population of vulnerabilities in the stable release of Chrome, its last and most widely available version, and counted duplicates only where there were two different parties involved. One of those parties can be Google itself, whether that is a disclosure from security team member or a product of the ClusterFuzz testing infrastructure.[3] More detail on these changes is available in Sections 3 and 6.

3 Speaking with members of the Chrome security team as well as the originator of ClusterFuzz, there was a consensus that ClusterFuzz is intended to discover vulnerabilities prior to malicious actors. Fuzzing is part of the range of discovery methods used to secure Chrome and thus counts as a reasonable source of discovery. – Conversation with the Authors, 26JUL17 and over email 12 and 15AUG17.

2. Background

Software vulnerabilities are flaws or features in code that allow a third party to manipulate the computer running this software. The level of design security in major commercial software products varies widely, from the vulnerability-rich history of Adobe's products to Apple's comparatively locked-down iOS operating system. Design insecurity is generally the result of poorly secured software, insecure programming languages, the growing complexity of commercial code bases, and simple human error, among a host of other causes. For example, a program that expects to retrieve a simple image file but fails to check the supplied file type might return an executable software program instead. The procedure to retrieve an image is intentional, but failing to check the file type allows a third party to manipulate it. The Love Letter virus of 2000 relied on the fact that Windows 2000 and XP hid known file extensions when reading file names from right to left. The virus file (LOVE-LETTER-FOR-YOU.TXT.vbs) hid itself by putting the executable file extension (.vbs) outside of a benign one (.txt), so Windows would only show .txt and the user would be none the wiser.[4] Vulnerabilities may also be introduced directly to hardware through compromises in chip design or manufacture somewhere along the supply chain.[5]

Not all vulnerabilities are created equal—some are easier to find than others and only a small number will provide access to the best-secured software. What this means is that not all groups looking for vulnerabilities are necessarily looking for the *same* vulnerabilities. An intelligence organization is likely to have the engineering and mathematics capacity to take low-value or difficult-to-use vulnerabilities and combine them into a working exploit. Less capable groups may have to wait until they find a vulnerability that can immediately be used to gain access to a computer system to develop a useful exploit.

4 Microsoft, "VBS.LOVELETTER Worm Virus" (Microsoft, January 29, 2007), http://support.microsoft.com/kb/282832.

5 Georg T. Becker et al., "Stealthy Dopant-Level Hardware Trojans," August 21, 2013, http://people.umass.edu/gbecker/BeckerChes13.pdf; Bruce Schneier, "How to Design — And Defend Against — The Perfect Security Backdoor," *WIRED*, October 16, 2013, https://www.wired.com/2013/10/how-to-design-and-defend-against-the-perfect-backdoor/.

The knowledge of a vulnerability's existence is valuable information, with similar properties to the location of buried treasure on a map or a secret told to a friend. The bug hunter's challenge is to choose who to whisper their secret to, along with proof-of-concept code that proves their secret is in fact true. This secret is a source of value and creates a dilemma: malicious actors who wish to gain access to a vulnerability are almost always willing to pay more than the software's original vendor—sometimes a great deal more.[6] Because a vulnerability is something embedded in a piece of software, a person who independently discovers it has no guarantee of being the only one who knows about its existence. Every passing day brings a higher probability that someone else working to find vulnerabilities in the same piece of software will stumble upon the bug, leading to rediscovery. This leads to an economy of buying and selling vulnerability information among criminal groups, companies, and governments.[7]

Value of Rediscovery

Vulnerabilities are information, which means there is no intrinsic reason they can't be held by multiple parties at once. Economics calls this characteristic being non-rivalrous. A rediscovered vulnerability may be the same as the original, or a closely related flaw deemed similar enough to be functionally identical. Rediscovery describes the likelihood that two independent parties will discover the same flaw in a piece of software. This is slightly different from a bug collision, which is when a vulnerability that had previously only been known to a single party enters the public domain. Our discussion about the rate of rediscovery assumes that the two groups that find the same bug are independent of each other.

6 A. Algarni and Y. Malaiya, "Software Vulnerability Markets: Discoverers and Buyers," *International Journal of Computer, Information Science and Engineering* 8, no. 3 (2014): 71–81.

7 T. J. Holt, "Examining the Forces Shaping Cybercrime Markets Online," *Social Science Computer Review* 31, no. 2 (September 10, 2012): 165–77, doi:10.1177/0894439312452998; Kurt Thomas et al., "Framing Dependencies Introduced by Underground Commoditization," 2015, http://damonmccoy.com/papers/WEIS15.pdf; Herr and Ellis - Ch.7 "Disrupting Malware Markets" in Richard Harrison and Trey Herr, eds., *Cyber Insecurity: Navigating the Perils of the Next Information Age* (Lanham, MD: Rowman & Littlefield, 2016), https://books.google.com/books?id=NAp7DQAAQBAJ&source. For more on the policy debate around which vulnerabilities governments should disclose, see Ari Schwartz and Rob Knake, "Government's Role in Vulnerability Disclosure" (Belfer Center, June 2016), http://www.belfercenter.org/sites/default/files/legacy/files/vulnerability-disclosure-web-final3.pdf; Fidler - Ch.17 "Government Acquisition and Use of Zero-Day Software Vulnerabilities" in Harrison and Herr, *Cyber Insecurity*.

Rediscovery is an important issue for the cybersecurity community, perhaps most prominently in the debate over how government should balance the choice to keep secret or disclose vulnerabilities to software developers. In the United States, the Vulnerabilities Equities Process (VEP) is the interagency process intended to make decisions about the disclosure of software vulnerabilities known by U.S. government agencies and organizations.[8] The key choice for the VEP is whether to keep a vulnerability, or vulnerabilities, secret or disclose these flaws to their developer, whereupon the government loses the opportunity to use the vulnerability (though the time it takes for a vulnerability to become useless after disclosure can vary dramatically). Measuring the cost of disclosure versus non-disclosure involves a range of factors, but an important factor is the likelihood that a vulnerability, kept secret from a vendor and unpatched, might be rediscovered by another party and used against U.S. citizens.[9] The answer is critical to determining the cost of non-disclosure of a vulnerability. If a vulnerability in the possession of the U.S. government is not likely to be discovered by another party, then the risk of keeping it a secret is lower than if the likelihood of rediscovery is high.[10]

However, there is more value in the question of rediscovery than just the VEP. Rediscovery can impact how the software security industry thinks about bug bounty programs, which pay researchers in exchange for disclosure of a software flaw. In paying for a vulnerability, companies expect that it can be patched (fixed). As these patches accumulate and users apply them, one of two things should happen: either these companies slowly reduce the total number of flaws in their codebase, or they find and fix enough old bugs to keep pace with new ones created as software is updated over time. Understanding the speed of rediscovery helps inform

8 Schwartz and Knake, "Government's Role in Vulnerability Disclosure"; Jason Healey, "The U.S. Government and Zero-Day Vulnerabilities: From Pre-Heartbleed to Shadow Brokers," *Columbia Journal of International Affairs*, November 2016, 4, https://jia.sipa.columbia.edu/sites/default/files/attachments/Healey%20VEP.pdf.

9 Mailyn Fidler and Trey Herr, "PATCH: Debating Codification of the VEP," *Lawfare*, May 17, 2017, https://www.lawfareblog.com/patch-debating-codification-vep.

10 The ShadowBrokers leaks could provide some insight into how often rediscovery of NSA vulnerabilities takes place. Out of five serious vulnerabilities included in the leak, only one had been previously discovered in use in the wild and patched (constituting rediscovery by another party against the NSA). This would produce a rediscovery rate of 20% but should be treated skeptically as there is no indication what proportion of the total NSA stock of vulnerabilities this leak constitutes or what the rate of this larger population might be. For more, see Rebekah Brown, "The Shadow Brokers Leaked Exploits Explained," *Rapid7 - Information Security*, April 18, 2017, https://community.rapid7.com/community/infosec/blog/2017/04/18/the-shadow-brokers-leaked-exploits-faq.

companies, showing how quickly a disclosed but unpatched bug could be rediscovered by a malicious party and used to assault the company's software. This information should drive patch cycles to be more responsive to vulnerabilities with short rediscovery lag, while allowing more time for those where the lag is longer. With additional work, rediscovery may also contribute to more accurate estimates of the density of vulnerabilities in software.

Academic research into the malware markets is also likely to benefit from better estimates of vulnerability rediscovery. Rediscovery impacts the lifespan of a vulnerability; the likelihood of its being disclosed to or discovered by the vendor grows with every instance of rediscovery. Just as one can compare a supermarket's need to renew its stock of bread versus that of salted herring, some vulnerabilities are more likely to "go stale" sooner, and thus be of little value.[11] Estimating rediscovery can help shed light on which types of vulnerabilities are more likely to decay relative to others, based on the frequency with which they are discovered by multiple parties.

Rediscovery and Previous Literature

Despite the importance of this issue, the academic record on rediscovery is relatively sparse.[12] A 2005 paper by Andy Ozment applied software reliability growth models to vulnerability discovery to gauge the total population of software vulnerabilities in the operating system BSD.[13] This paper also addressed rediscovery, using a collection of vulnerability bulletins from Microsoft between 2002 and 2004 to catalogue when the company credited more than one disclosing party. Ozment found an average rediscovery rate of just under 8%, aggregated over different types of software, including operating systems, applications, and supporting libraries.

11 A vulnerability can rise in value if integrated with many others as part of an exploit kit, a mechanism to deploy malicious software using dozens of vulnerabilities, or if few targets apply the patch that fixes the corresponding flaw.

12 There is an ample literature on vulnerability discovery that deals with an important but slightly different question from this paper's focus on *re*discovery. For a detailed literature review, see Fabio Massacci and Viet Hung Nguyen, "An Empirical Methodology to Evaluate Vulnerability Discovery Models," *IEEE Transactions on Software Engineering* 40, no. 12 (2014): 1147–1162.

13 Andy Ozment, "The Likelihood of Vulnerability Rediscovery and the Social Utility of Vulnerability Hunting," 2005, http://www.infosecon.net/workshop/pdf/10.pdf.

Writing in 2013, Finifter et al. looked at the relative cost efficiency of vulnerability reward programs against directly employing security personnel. Looking at Firefox and Chrome, they found that most vulnerabilities are reported from within firms, though by 2012 this trend had shifted for critical vulnerabilities in Chrome and more were reported from outside the company. In a small section looking at rediscovery, the group's paper calculated a mean rediscovery rate for Chrome of 4.6% and provided anecdotal evidence of similar rates in Firefox.[14]

A collaboration between the bug bounty company HackerOne and researchers at Harvard and MIT produced a system dynamics model of the vulnerability discovery and stockpiling process.[15] As part of this work, the group presented results of a random discovery simulation at RSA that showed a 9% rediscovery rate for immature software and less than 1% for "hardened" or more mature codebases. Unfortunately, no formal paper describing the methodology and data employed in this study has yet been published.

In each of these instances, the question of rediscovery was tangential to a different debate. Ozment's work was a response to Eric Rescorla's contention that the long-term utility of vulnerability discovery and patching was low.[16] Finifter and his group were comparing the efficacy of bounties to that of hiring security talent as full-time employees, and the HackerOne estimate of rediscovery came in the context of discussing the relative density of vulnerabilities in old vs. new software. In each of these, there was little actual data available to judge the rate of rediscovery and other potentially interesting characteristics.

Most recently, a 2017 study from the RAND Corporation used a small private dataset to evaluate the nature and behavior of zero-day

14 Matthew Finifter, Devdatta Akhawe, and David Wagner, "An Empirical Study of Vulnerability Rewards Programs," in *USENIX Security*, 2013, https://www.usenix.org/system/files/conference/usenixsecurity13/sec13-paper_finifter.pdf.

15 Katie Moussouris and Michael Siegel, "The Wolves of Vuln Street: The 1st Dynamic Systems Model of the 0day Market" (RSA, 2015), https://www.rsaconference.com/events/us15/agenda/sessions/1749/the-wolves-of-vuln-street-the-1st-dynamic-systems.

16 Eric Rescorla, "Is Finding Security Holes a Good Idea?," *IEEE Security and Privacy* 3, no. 1 (January 2005): 14–19.

vulnerabilities—those used by attackers before the vendors learn about them.[17] The study found that over the span of a year, on average only 5.76% of vulnerabilities were rediscovered in the public domain, while for 90 days or less the figure was less than 1%. This is lower than our findings which saw annualized rates of between 10.8% and 21.9%.[18] While the two papers are scoped to different ends, much of the distinction is likely attributable to differences in data sources and methodology. For more on these differences, see Section 6.

3. Methodology and Data

This paper addresses a gap in the literature by integrating vulnerabilities reported in several different codebases, including the browsers Firefox and Chrome, the open source project OpenSSL, and the Android operating system to generate estimates of vulnerability rediscovery and related measures such as rediscovery lag.[19] The goal in selecting these codebases was to cover more than one software type, span multiple vendors, and have the best possible access to complete data.[20]

Counting Duplicate Vulnerabilities

Measuring rediscovery is difficult because once the original vulnerability is disclosed and made public, there is little incentive for anyone to come forward and make a new disclosure about the same vulnerability, except where to do so might result in reputational rewards. The dataset collected here, and all those that look at disclosure records, don't measure discovery directly. Instead, this data captures disclosure as a proxy for discovery. This

17 Lillian Ablon and Andy Bogart, "Zero Days, Thousands of Nights" (Santa Monica, CA: The RAND Corporation, 2017), https://www.rand.org/content/dam/rand/pubs/research_reports/RR1700/RR1751/RAND_RR1751.pdf.

18 OpenSSL is the outlier, with 57 bugs and 2 duplicates, this library's rediscovery rate is 3.4%. We explore possible explanations for its variance with the other three software in the following section.

19 A codebase is the collection of code used to develop a piece of software, including both current and previous versions.

20 All the data we used is available in our GitHub repository (https://github.com/mase-gh/Vulnerability-Rediscovery), and we continue to work to expand this dataset to include closed source software and other open source projects.

limits the "rediscovery window" to capture rediscovery for each vulnerability record to the period between an initial vulnerability disclosure and public notice of the bug's existence.

Limiting our data to this rediscovery window has some benefit as well. Looking at rediscovery, there is a reasonable assumption that over a long enough period any vulnerability will be discovered multiple times. Because the window in which we can observe rediscovery is effectively limited to the period between initial disclosure and when a patch is made publicly available, there is a natural time constraint. This is built on Ozment's method of counting multiple credited discoverers, a record-keeping process that would end with a patch being made available to the public.

Within this rediscovery window, our method of tabulating the discovery of a rediscovered vulnerability looks for two criteria: are there multiple parties given credit for independently disclosing the same vulnerability, and/or has the bug been marked a duplicate and merged with another from a different discloser? For OpenSSL, we only observed instances of credit being given to multiple parties, as that database did not include duplicates or merge information. More detail on the specific method of counting duplicates for each piece of software can be found below.

To calculate rediscovery, we measure the total number of vulnerability records with duplicates as a proportion of all vulnerability records for a piece of software. In a given period, if there are ten different vulnerability records and one of those records has received duplicate disclosures, then only one vulnerability record has a duplicate. As a result, the rediscovery rate is 1 in 10, or 10%.[21] In our estimates, we collect the total population of vulnerabilities and sample those of high or critical severity to measure this rediscovery rate. Some of those duplicate reports come from software vendors. While rediscovery can't take place between two discoveries from the same organization, the same vulnerability found within a vendor and then disclosed from external party would certainly count.

21 This 10% figure shows the proportion of vulnerabilities that are rediscovered and not the total number of duplicates. If that same vulnerability was rediscovered 10 more times, for a total of 20 vulnerability disclosures, the rediscovery rate is unchanged in this methodology. This is not to say these additional duplicates aren't of interest; we address their frequency and significance in *Multiple Rediscovery* in Section 4 below.

The other two measures presented in this paper, rediscovery over time and rediscovery lag, are derived from the same data. The likelihood of rediscovery appears to increase in the months after a vulnerability's initial disclosure, eventually leveling off within a few months. Rediscovery over time measures this rate of change. Rediscovery lag measures the time between an original disclosure and any subsequent duplicate disclosures; e.g., the time between $\text{Disclosure}_{\text{Original}}$ and $\text{Disclosure}_{\text{DuplicateA}}$ is X and time between $\text{Disclosure}_{\text{DuplicateA}}$ and $\text{Disclosure}_{\text{DuplicateB}}$ is Y. Thus, the rediscovery lag for Duplicate A would be X while the lag for Duplicate B would be X+Y.

Coding and Data Sources

In each source of data for vulnerability rediscovery, we used only vulnerabilities of high or critical severity to improve the quality of our data and impact of our analysis. The exception for this is OpenSSL where we used all critical, high, and medium severity records (more on this below). Software bugs are generally given a severity score to help organize them and prioritize which need to be fixed first. The definitions of high and critical severity vary somewhat between organizations but generally settle on critical as including anything that allows the execution of arbitrary code, while high covers most instances where an attacker could manipulate software functions or operate without restriction if local to the targeted computer. For example, Google defines severity as:

- Critical: "issues allow an attacker to run arbitrary code on the underlying platform with the user's privileges in the normal course of browsing."

- High: "...vulnerabilities allow an attacker to execute code in the context of, or otherwise impersonate other origins. Bugs which would normally be critical severity with unusual mitigating factors may be rated as high severity."[22]

22 The Chromium Projects, "Severity Guidelines for Security Issues," *For Developers*, https://sites.google.com/a/chromium.org/dev/developers/severity-guidelines.

By constraining this dataset to high and critical vulnerabilities, the paper offers analyses based on the most impactful software flaws. This means that the dataset comprises only a subset of the total population of vulnerabilities but emphasizes those most critical to developers and policymakers. This has also generally improved the quality of data, as record keeping appears to be most detailed for these most important bugs. We intend to expand to medium- and low-severity bugs, and their equivalent naming conventions across different vendors, in future work.

In collecting data for this project, we spoke with more than a dozen vendors and open source projects. Despite helpful conversations with security and privacy team members at Apple, Microsoft, Cisco, and several others—we were unable to obtain anonymized vulnerability disclosure records from any of the major software vendors. Google cooperated to provide access to an internal database of Android disclosures but was unable to share data from proprietary software like the Nest line. We selected OpenSSL as an example library but also considered OpenSSH and ntpd, both of which were left out of the analysis because of the limited vulnerability record information available. As this work, and that of other groups continues, we hope access to these sources of vulnerability record information will improve.

Data for the software used in this paper came from four sources that we integrated into a single dataset. There are numerous challenges in counting vulnerabilities, and the variable state of record keeping discovered across vendors and open source projects only underlines this. By limiting our collection to these high- and critical-severity vulnerabilities, the dataset is made less generalizable but more reliable.

Chrome

The Chrome dataset has been revised from its original version.[23] The Chromium data was scraped from the Chromium bug tracker and manually

23 Original paper with this analysis available here: https://github.com/mase-gh/Vulnerability-Rediscovery/blob/master/Vulnerability%20Rediscovery.pdf

evaluated to identify duplicate disclosures.[24] Chromium is an open source software project whose code constitutes most of the Chrome browser; Google adds a few additional features, such as a PDF viewer, but there is substantial overlap, so we follow on other research and treat this as essentially similar to Chrome.[25] In our previous approach, we misunderstood the inconsistency of the metadata in this database, including how the label "duplicate" was attached to bug reports. In some cases, both duplicates were generated by Google's testing infrastructure or were duplicates submitted by the same person and thus not an instance of rediscovery. In other instances, these duplicates matched our coding of a duplicate—different individuals disclosing the same bug. In Table 1, the Total Population value is the total count of vulnerabilities regardless of severity in the Chrome stable release channel between 2009 and 2017. The sample of this population we used to measure rediscovery is indicated in the Vulnerabilities" column, with the total count of duplicates under Duplicates.

ClusterFuzz

Some of the duplicates in the Chromium database are automatically generated. With that in mind, we should explain ClusterFuzz. ClusterFuzz is the Google Chrome team's ensemble fuzzer, a collection of programs (fuzzers) that throw junk data at a software program to cause it to crash. Those crashes, recorded as crash reports, are collected and evaluated to see if they represent a security vulnerability. What makes ClusterFuzz interesting is that it can do much more than just generate these crash reports. The Chrome team also uses ClusterFuzz to assign the security severity rating for vulnerabilities, essentially the priority they should be dealt with according to what impact they could have, including all bugs submitted from outside disclosers. Most importantly, ClusterFuzz can submit all the bugs it generates automatically to the Chromium database. There is a manual review of these but this is considered a rubber stamp on the process.[26]

---

24 The Chromium Projects, "Chromium Bug Tracker," *Bug*s, current, https://bugs.chromium.org/p/chromium/issues/list?can=1&q=label%3ASecurity_Severity&sort=security_severity&colspec=ID+Component+Summary+Security_Severity+Reward+Reporter+Status&x=m&y=releaseblock&cells=ids.

25 Others have previously made this same choice, including Finifter, Akhawe, and Wagner, "An Empirical Study of Vulnerability Rewards Programs."

26 Conversation between a member of the Chrome security team and the authors, 21JUL17

Because ClusterFuzz can identify, assign severity, and automatically submit bugs to the Chromium database, there are duplicates in the Chromium database that were generated by ClusterFuzz. This is more common with Chrome's developmental releases—those updated nightly or weekly and used primarily by the testing community. In parsing the Chromium database, some of the vulnerabilities labeled as duplicates had been generated by the same source, and thus not good examples of rediscovery. However, many were discovered by different parties, including where ClusterFuzz was one of the parties.

Many instances of rediscovery take place using similar or even the same tools and techniques. That the Chrome team used a common tool infrastructure, not even necessarily the same fuzzer, is irrelevant from our standpoint. An open source tool, such as a fuzzer, can be used by different parties to find the same vulnerability. This is nevertheless considered an instance of rediscovery. Simply using a fuzzer doesn't obviate the chance of rediscovery and, indeed, a tool being widely available makes it more likely to be included in the discovery infrastructure of malicious parties. This is exactly what rediscovery looks like—instances where the same potential vulnerability is discovered by two different parties. We count instances where ClusterFuzz and an external discloser discover the same vulnerability because fuzzing is part of Google's security infrastructure and Google is a valid party for rediscovery. We do not count instances where ClusterFuzz rediscovers the same vulnerability previously discovered by itself, as this does not meet the two independent actors criteria.

Revising Chrome

As in our previous analysis, we limited our population of vulnerabilities only to high- and critical-severity bugs. In this revision we also constrained our population to only those bugs from Chrome builds in the Stable release channel. These are versions of Chrome that have the widest possible release and so are a) exposed to the most people for discovery, b) theoretically the most secure against discovery, having been pushed through three progressively more public release channels, and c) should see the lowest rates of bug churn—where vulnerabilities are identified and patched in short periods of time.

Because the Chromium database duplicate labels were assigned according to a system different from our own, we went through every vulnerability marked as a duplicate and validated it by hand before parsing the full database again (this parsing script is available on our GitHub page). Our revised coding framework specified that two disclosures could only be counted as rediscovery where they were submitted by different parties. We also stepped through each bug coded as a duplicate manually, which resulted in dropping additional bugs from our count of duplicates. For example, bug 95072 (CVE-2011-2877) is a vulnerability in how Chrome parses a text file form. It has multiple duplicates in the Chromium database, all submitted from different disclosers. These two duplicates could be counted as valid rediscovery; however, the first, 96958, was diagnosed as being related to another piece of software unrelated to Chrome and the second, 99232, was submitted but couldn't be reproduced. This is a conservative way to parse this data, so we may be unnecessarily rejecting legitimate duplicates by not manually checking through the stack traces for each bug report.

This approach may also fail to capture legitimate duplicates without multiple disclosers. For example, bug 119281 (CVE-2011-3074) was first disclosed on 21MAR12 and then disclosed by a second party on 28MAR12. This alone was rediscovery, but meanwhile, the original disclosure was noticed by a member of the Chrome security team and identified in the original bug's comment history on 4APR12 as a bug that had been discovered internally by Google as well.[27] Without the second disclosure on 28MAR12, this would have been dropped from the dataset.

Below are several examples of what we counted as rediscovery:

- Bug 71788 (CVE-2011-1196) was independently discovered by Chrome community member SkyLined on 3FEB11 and David Westin of Microsoft on 14FEB11.

- Bug 67303 (no CVE assigned) was independently discovered by David Warren of Software Engineering Institute/CERT

27 The Chromium Projects, "Chromium Bug Tracker," *Bugs*, current, https://bugs.chromium.org/p/chromium/issues/detail?id=119281#c16.

on 17DEC10 and Chrome community member SkyLined on 27DEC10.

- Bug 449739 (CVE-2015-1251) affects the SpeechRecognitionDispatcher and was discovered internally by Google on 17JAN15. It was then disclosed on 5FEB15 by the Zero Day Initiative (ZDI), via either by internal discovery or disclosure from a third party to ZDI.

Each bug was disclosed by two different parties, had the original and/or the duplicate record labeled as high or critical severity, and was present in the Stable release channel for Chrome. This portion of the dataset comprises **1,798 vulnerability records**, of which there are **195 records with at least one duplicate.**

Android

The Android data we sourced was developed in collaboration with a member of the Google Product Security Response team. We were provided access to data from Google's tracking systems for Android vulnerabilities: the public facing Issue Tracker as well as an internal-use-only platform. The internal platform used by Google employees tracks disclosures from within the company as well as some from outside, while the public site records disclosures from outside the company. This data provides a glimpse into rediscovery rates for Android over a 17-month time frame between July 2015 and November 2016. A member of Google's Product Security Team downloaded bugs from both tracking systems, then correlated them to remove identical records that existed in both systems.[28] This individual then added a CVE ID for all records, to replace the internal tracking IDs. Instances of rediscovery—duplicate disclosures—were coded when a vulnerability record in the public site was noted as a duplicate and linked to a report credited to a different researcher than the original. Duplicates were also coded if two bugs were merged together. The Android portion of the paper's dataset comprises **352 vulnerability records**, of which **77 records had at least 1 duplicate**.

28 This is possible because Google maintains both respective tracking systems' vulnerability ID value for all records on both platforms.

There is no indication in the Android data of the sort of automatic generation of duplicate vulnerabilities that took place with Chrome. Several of the duplicates are associated with the so-called Stagefright Bug, originally disclosed at Black Hat in 2015. This bug allowed an attacker to hijack an Android device simply by sending a crafted media message to the phone with no user input required. Of the seven Stagefright bugs originally disclosed by Zimperium, only one is included in this dataset.[29] [30] Many of the related Stagefright bugs are not disrupted by the patch for this original bug, and several were rediscovered, and thus certainly constitute independent, critical-severity vulnerability rediscoveries. For example, CVE-2015-6600,[31] a vulnerability in the Stagefright library in Android, exploitable through a specially crafted media file, was rediscovered twice.[32]

One critique of this data is that it does not correspond with the transparency of the other data sources used for this paper—that complete transparency would require more than is hosted in our GitHub repository. With respect to data access, all the bugs used in our analysis are available on GitHub with CVE codes and their respective dates of discovery and, where applicable, rediscovery. While the same level of transparency as with the Chrome and Firefox data would be ideal, Google's issue tracking and mitigation system is not structured the same way as Android's. Some Android vulnerability disclosure records are only available through an employee-accessible internal database. Our access to this internal tracker was one step removed; this was deemed a reasonable condition of receiving access to this data.

There are means to further verify some of these duplicates, including using Android security bulletins and individual researcher publications. One example is CVE-2015-6639, a vulnerability allowing attackers to run code in the Qualcomm Secure Execution Environment (QSEE). 2015-6639 was initially discovered by Gal Beniamini, but also discovered within

29 CVE 2015s -1538, -1539, -3824, -3827, -3828, -3829 are not included; -3826 is present

30 Zimperium, "Experts Found a Unicorn in the Heart of Android," *Zimperium Mobile Security Blog*, July 27, 2015, https://blog.zimperium.com/experts-found-a-unicorn-in-the-heart-of-android/.

31 National Institute of Standards and Technology, "National Vulnerabilities Database", *Vulnerabilities* current, https://nvd.nist.gov/vuln/detail/CVE-2015-6600.

32 Looking for bugs where others have already gone may not win you praise from Pastor Laphroaig but it doesn't impact this analysis.

Qualcomm and disclosed to Google directly.[33] This duplicate disclosure was made public on the corresponding Android Security Bulletin. [34] Evaluating the credits on several Android patch announcements, unnamed credit was also given to members of various security teams, including Google's Project Zero. These individuals were likely often the source of some duplicates noted on the internal issue tracker.[35]

Firefox

The Firefox dataset was scraped from records in the Bugzilla bug tracker for Firefox and related software dependencies.[36] From a total population of 1,119 labeled bugs, we sampled only vulnerabilities labeled as high-priority or critical in the Severity field in Extended Service Release (ESR) advisories for Firefox between 2012 and 2016. This sample comprises **472 vulnerability records** and **66 records with duplicates**. Firefox presented a challenge in how to create a working subset of only those vulnerabilities from the total pool of bug records. Firefox has nightly builds—new versions of the codebases with small changes or trial features—and many of the vulnerabilities discovered in Firefox are found there and fixed immediately. This dataset does not include these vulnerabilities, since they are never exposed to the public as part of an Extended Stable Release (ESR), and are thus unlikely to represent bugs that might be independently rediscovered. By counting only bugs from ESRs, we could obtain a subset of vulnerabilities that were exposed for discovery and exploitation by all users. In Bugzilla, each vulnerability record has a report page, with a Tracker subsection. For some vulnerabilities, that Tracker subsection contains a Duplicates field with a record of associated bug codes and their status. Those records with data in the Duplicates field were what was coded as duplicates.

As with Android, nothing in our discussion with Firefox's security team, or subsequent review, indicates concerns with the Bugzilla database such

33 Gal Beniamini, "QSEE Privilege Escalation Vulnerability and Exploit (CVE-2015-6639)," *Bits, Please!*, February 5, 2016, http://bits-please.blogspot.com/2016/05/qsee-privilege-escalation-vulnerability.html.

34 Android Open Source Project, "Nexus Security Bulletin – January 2016", *Security* January, 2016, https://source.android.com/security/bulletin/2016-01-01.

35 Android Open Source Project, "Nexus Security Bulletin – October 2015", *Security* October, 2015, https://source.android.com/security/bulletin/2015-10-01.

36 Mozilla, "Bugzilla," *Bugzilla@Mozilla*, https://bugzilla.mozilla.org/buglist.cgi?quicksearch=ALL%20kw:sec%20cf_status_firefox_esr31%3Afixed%2Cverified.

as those encountered for Chromium. This includes accounting for the behavior of Mozilla's internal fuzzing infrastructure, FuzzManager. Unlike ClusterFuzz, FuzzManager does not automatically assign security ratings and, importantly, does not automatically log reports into Bugzilla. In expanding the data to cover 2017, we have filtered any duplicates that do not meet the coding criteria of high or critical severity and different disclosers. This has changed the aggregate rediscovery rate from 17.1% to 14%. Looking through the dataset, we can quickly find several instances of high-security-impact vulnerabilities. One such was CVE-2014-1551, a high-severity bug that was discovered by three different people and could allow an attacker to execute arbitrary code through several different versions of Firefox and the Thunderbird mail application (which shares code with the browser).

OpenSSL

Our working sample from OpenSSL comprises all vulnerability records of critical, high, or medium severity from 2014 to 2016 for the OpenSSL project.[37] The start time for this window is the disclosure of the Heartbleed vulnerability. OpenSSL's records on vulnerability disclosures are unreliable prior to Heartbleed. This is not a criticism of the project, an open source endeavor which supported as much as two thirds of all web traffic in 2014 and operated on a shoestring budget until just a few years ago.[38] OpenSSL did not fail to record bugs received or mitigated, but rather the disclosure records suffered from inconsistent record keeping of non-critical information such as additional duplicate disclosures. This inconsistency is a common issue across vulnerability disclosure programs, and is particularly challenging in the open source community. We made note of this in the original paper as a result of conversations with members of the open source projects like OpenSSL and OpenSSH.

There is reason to expect inconsistent record keeping of duplicate disclosures with the OpenSSL data, and in fact there is also evidence of it. The

37 OpenSSL, "OpenSSL Vulnerabilities," *News/Vulnerabilities*, https://www.openssl.org/news/vulnerabilities.xml.

38 Dan Goodin, "Critical Crypto Bug in OpenSSL Opens Two-Thirds of the Web to Eavesdropping," *Ars Technica*, Aptil 2014, https://arstechnica.com/information-technology/2014/04/critical-crypto-bug-in-openssl-opens-two-thirds-of-the-web-to-eavesdropping/.

2014 Heartbleed bug was a canonical example of rediscovery—present in the OpenSSL code for years, then independently discovered by two parties, Neel Mehta and Finnish security company Codenomicon, within days of each other.[39] Yet on OpenSSL's vulnerability page, only Mehta is credited. We included OpenSSL in the paper with the aim of broadening the dataset beyond mainstream consumer software. This portion of the dataset comprises **57 vulnerability records** of which **2 have duplicate disclosures.**[40] In OpenSSL, vulnerabilities are noted as a duplicate if they credit two separate disclosers for the bug. Records with an "and" between credited reporters are coded as a collaboration and thus are not duplicate disclosures. Records with an "&" between reporters are coded as independent discoveries and thus duplicates.

39 Bruce Schneier, "Simultaneous Discovery of Vulnerabilities - Schneier on Security," February 25, 2016, https://www.schneier.com/blog/archives/2016/02/simultaneous_di.html.

40 Underlining the record-keeping issue—neither of these disclosures includes the Heartbleed bug, which was discovered twice within the span of a few days and reported in April 2014.

Table 1—Dataset Summary

Source	Date Range	Total Population	Vulnerabilities	Duplicates	Rediscovery Rate
Google—Chrome	2009–2017	2,323	1,798	195	10.8%
Google—Android	2015–2016	682[41]	352	77	21.9%
Mozilla—Firefox	2012–2016	1,119	472	66	14.0%
OpenSSL	2014–2016	85	57	2	3.5%
Total	**2009–2016**	**4,209**	**2,679**	**340**	**12.7%**

The above table summarizes the four sources for this paper's dataset, covering more than 2,600 vulnerability records over nine years from four different software projects.

- Source—the software these vulnerabilities come from, explained in detail above
- Date Range—the time range for the population of vulnerabilities and our sample
- Total Population—the total number of vulnerabilities of any severity level available for the specified release type (e.g., Stable release in Chrome) from each of the data sources for the specified date range
- Sample Vulnerabilities—our sample of high and critical vulnerabilities, a subset of the total population of vulnerabilities for the source software
- Sample Duplicates—the number of vulnerability record from our sample which had duplicates; see above for more detailed explanation of what constitutes a duplicate for each source
- Rediscovery Rate—the proportion of vulnerabilities from each source with at least one duplicate disclosure.

41 * Because of our inability to directly access Google's records for the total population of Android vulnerabilities, we use here the total number for Android reported to the National Vulnerability Database in the specified date range

4. Analysis

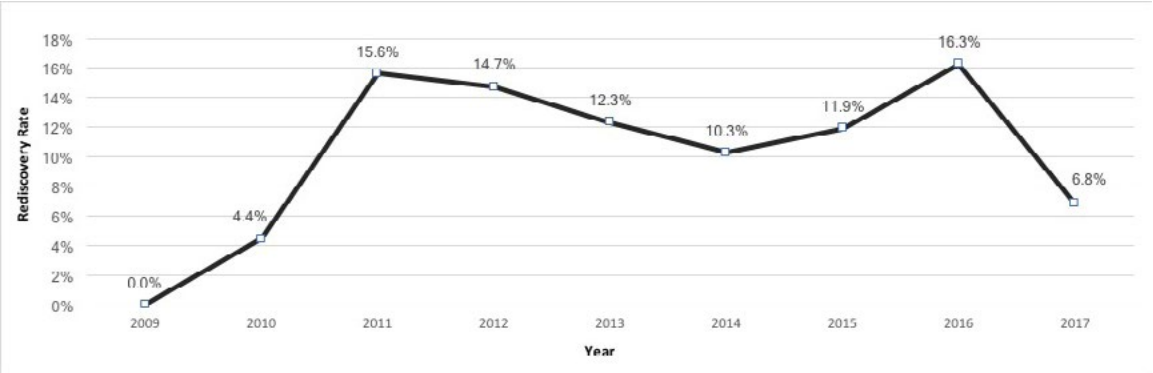
Our analysis covers vulnerabilities in a range of software types, including standalone applications like Chrome and Firefox, and the library OpenSSL. The first section of analysis takes all this software together, producing an aggregate measure of vulnerability rediscovery. Following this is an analysis of multiple rediscovery (where there are multiple duplicate bugs), evaluating trends in specific codebases, and then an analysis of rediscovery lag, the time between initial disclosure and the first duplicate report. The final subsection evaluates rediscovery over time. Each section draws from all or part of the dataset, with explanations for the use of subsets where appropriate.

Vulnerability Rediscovery in the Aggregate

Vulnerabilities in the nine-year span of this dataset see an aggregate 12.7% rate of rediscovery without regard for time or software. This is higher than previous open-source estimates, which ranged from 6.84% in early empirical work to 9% in more recent simulations.[42] Figure 1 below charts the annualized discovery rate over the entire dataset, which varies between 0% and 16.3%, depending on the year. The negligible rate of discovery in 2009 may well be attributable to the relative immaturity and small size of the discovery community looking at Chrome; its initial release was in September 2008. It should be noted that in Figure 1 and all graphs following, 2017 data is presented to capture as much of the behavior of interest as possible, but it is necessarily incomplete, given this paper's date of publication.

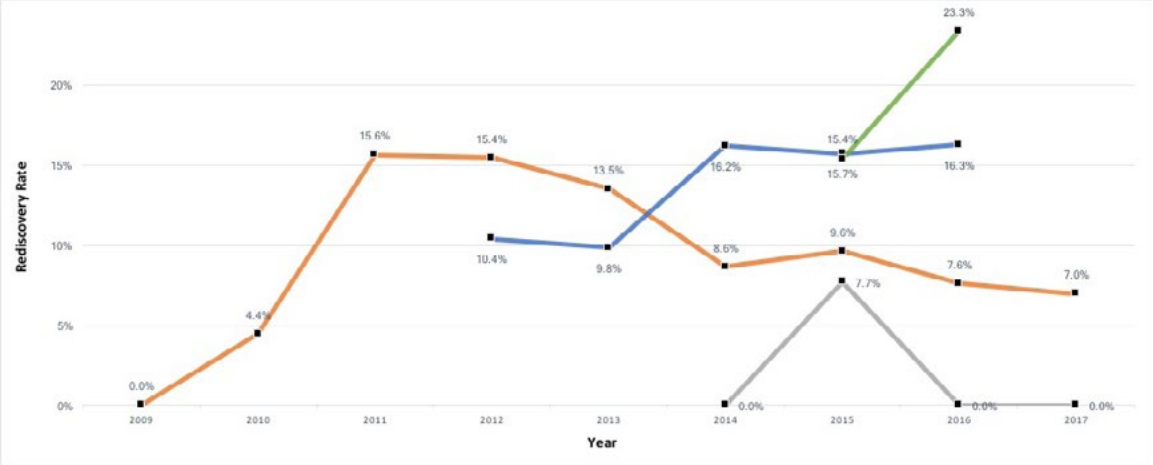
42 Ozment, "The Likelihood of Vulnerability Rediscovery and the Social Utility of Vulnerability Hunting"; Moussouris and Siegel, "The Wolves of Vuln Street."

Figure 1—Full Dataset: Aggregate Rediscovery Rate Over Time[43]



Some of this variability is driven by the different rediscovery rates of each codebase. Figure 2 breaks out each piece of software with its corresponding rediscovery rate per year.

Figure 2—Full Dataset: Rediscovery Rates by Software and Year[44]



The rates of rediscovery vary between software but also over time. From 2010 on, the bottom end of the range of values for the three largest programs, Chrome, Firefox, and Android, is 4.4%, while the upper end is 23.3%. This variation shows some of the importance of software type in determining rediscovery rates.

43 Figure 1: Calculated with all the software in the dataset in aggregate, by taking total number of vulnerability records with duplicates and dividing that by the total number of vulnerability records in each year.

44 Figure 2: Calculated for each software individually, by the same duplicates over total process as Figure 1.

Multiple Rediscovery

This section looks at the rate of rediscovery by codebase and the phenomenon of multiple rediscovery, where more than two parties disclose the same vulnerability. While we are not able to control for each vendor characteristic individually—for example, the difference between bug bounty payouts or secure coding practices—this section demonstrates that there are relatively consistent trends in multiple rediscovery rates between the vendors in our sample. Across this paper's entire dataset, where rediscovery did take place, one duplicate was the norm, though some vulnerabilities saw as many as four, five, and even one case of 11 duplicate disclosures. Figures 3, 4, and 5 describe this multiple rediscovery, putting it in context across Firefox, Chrome, and Android. Each pie chart has three values:

- No Duplicates: vulnerabilities for which there was only a single disclosure
- One Duplicate: vulnerabilities with an initial disclosure and one duplicate
- Two or More Duplicates: vulnerabilities with an initial disclosure and more than one duplicate. Note that this counts all vulnerability records with more than one duplicate, not the total of all these duplicates together.

We break the three biggest codebases out below, to show trends and differences in rediscovery, including distinctions between vulnerabilities with one duplicwate and those with two or more.

Figure 3—Chrome Vulnerability Rediscovery[45]

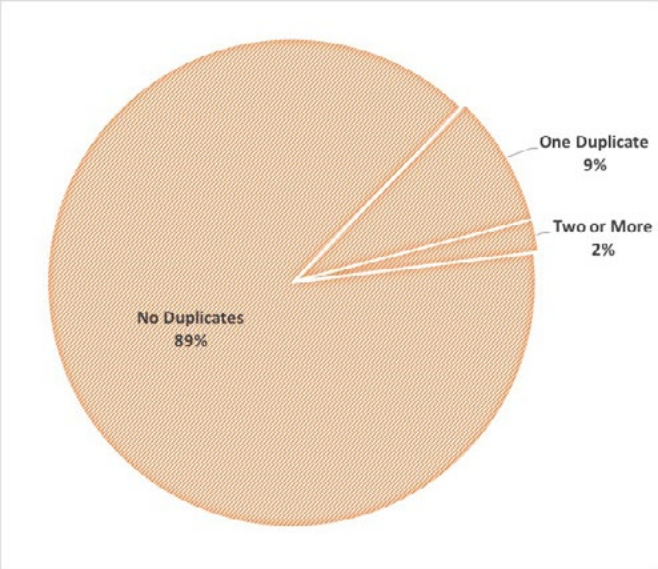
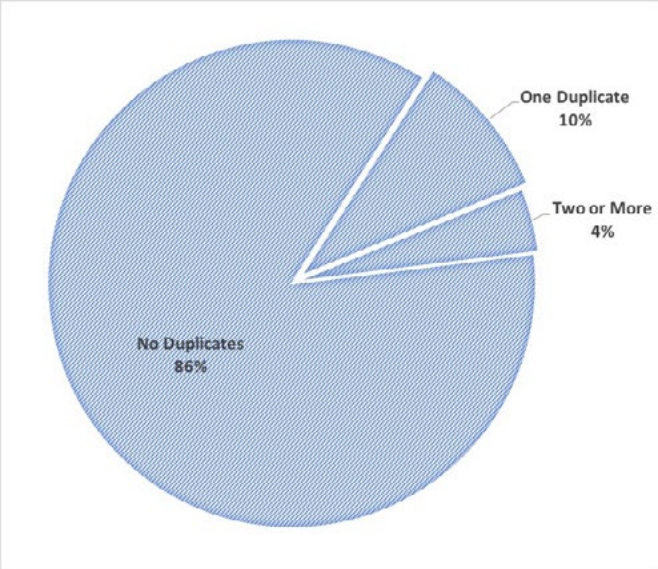
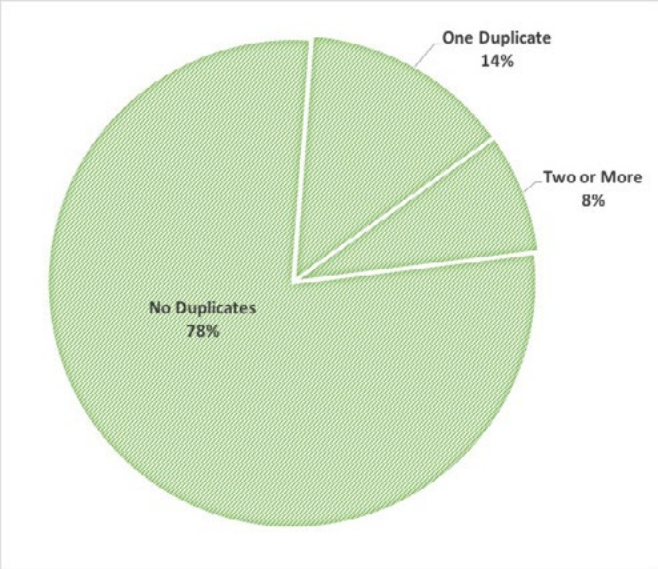


Figure 4—Firefox Vulnerability Rediscovery



45 Figures 3/4/5: Calculated by taking all solo duplicates (all records with duplicates—those with multiple duplicates) and the multiple duplicates (any record with multiple duplicates) as a proportion of total vulnerabilities with no duplicates (total vulnerability record—duplicates).

Figure 5—Android Vulnerability Rediscovery



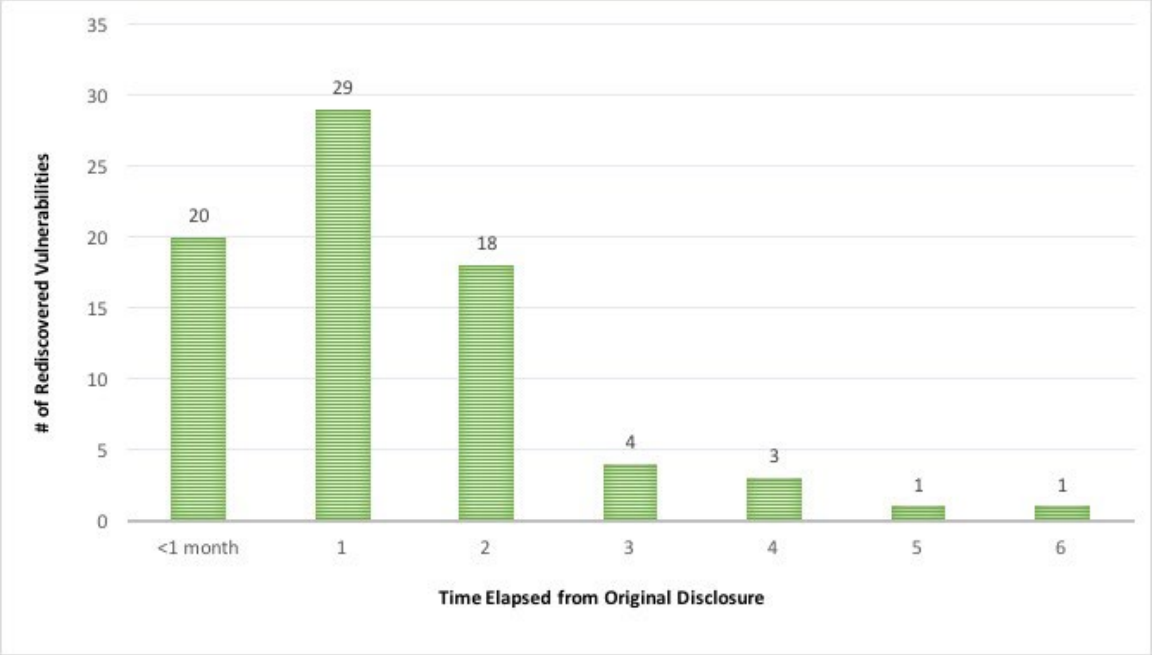
Multiple rediscovery increases for all three in line with overall rediscovery rates, which should be expected if vulnerability rediscovery is driven by largely the same factors as the initial vulnerability discovery. This correlation between rediscovery and multiple rediscovery may also be because newly rediscovered vulnerabilities are comparatively shallow (easier to find), and thus more likely to be disclosed by multiple parties. The ratio of multiple discovery varies somewhat; while about 20% of Chrome vulnerabilities with duplicates are rediscovered multiple times, both Firefox and Android hover around 30%. Some of this may be attributable to differences in the composition of the discovery communities between the three, although it is interesting to see the distinction between the two browsers.

Vulnerability Rediscovery Lag

Rediscovery can be a useful metric, but we should also consider the element of time and its impact on subsequent additional discoveries of the same vulnerability. This section looks at the time between original and duplicate disclosure: the rediscovery lag. This section uses the Chrome and Android data both to highlight differences between software types and to make use of software which, while open source, is primarily engineered and distributed by the same vendor. Rediscovery lag is calculated by taking the absolute time difference between original vulnerability disclosure and

first disclosure.[46] This section presents the Android data sorted into discrete monthly categories, as they were generated and shared with us. For Android, the average time between initial discovery and rediscovery was just under 1.5 months. Figure 6 shows a distribution of Android rediscovery lag for all duplicates, covering 77 distinct vulnerabilities.

Figure 6—Android Rediscovery Lag[47]



The disclosure of most duplicate vulnerabilities in Android takes place months later than the original, while only 20 occur in the same month. This challenges the notion of rediscovery as being largely simultaneous discovery, and suggests instead a more independent activity.

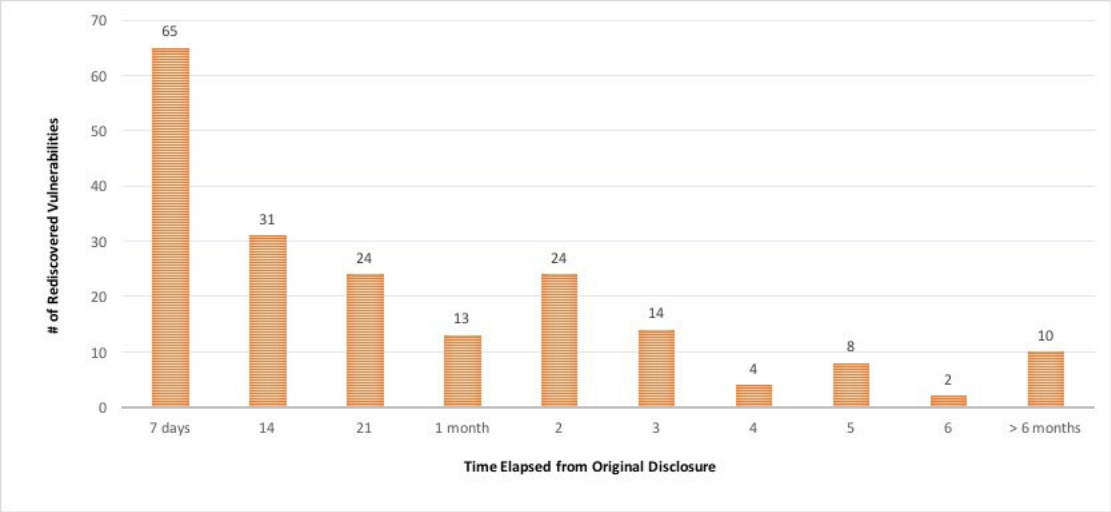
Figure 7 shows the time from original to first rediscovery for a subset of the vulnerabilities in Chrome. In Chrome, the largest portion of rediscovered vulnerabilities are disclosed in the first week. The chart shows how the proportion of duplicates disclosed decreases over time, with only 10 out of a total of 195 disclosed more than six months after the original disclosure.

46 In a previous version of this paper we included lag for multiple disclosures where records contained them. To simplify the presentation and allow these charts to be compared directly to those that follow, displaying rediscovery rates over time, we have limited these to only the lag between original and first duplicate disclosure.

47 Figures 6 and 7: Calculated by taking each vulnerability record and counting the days between the original disclosure and first duplicate disclosure. These durations are then displayed as a histogram with varying bin lengths showing the distribution of rediscovery lag (time between original and first duplicate disclosure).

There is more transparency in the Chrome issue tracking process without the internal Google issue tracker as for Android. This might increase awareness of other potentially similar bugs, driving duplicate disclosure close to originals.

Figure 7—Chrome Rediscovery Lag



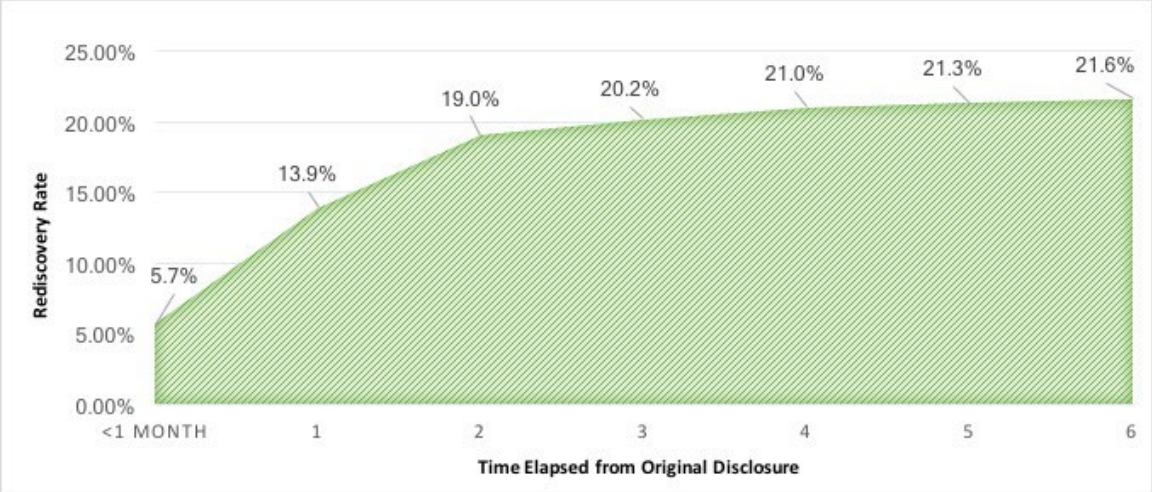
One factor that could help explain these differences in rediscovery rate between software packages is the length of time between the initial vulnerability report and when a patch was issued. This patch window is the capture period for our data so a smaller window could bias results lower, while a longer window could bias results higher. While this paper doesn't evaluate causes for rediscovery rates, this influence of varying patch window length is an issue ripe for further work

Rediscovery Over Time

While the rediscovery lag can give some picture of how long it takes to find a vulnerability, another way to consider the problem of what proportion are likely to be rediscovered within a given period of time. For Android, that rediscovery is slower to increase but ends up higher, topping out at 21.6% by six months from the original disclosure, as seen in Figure 8.

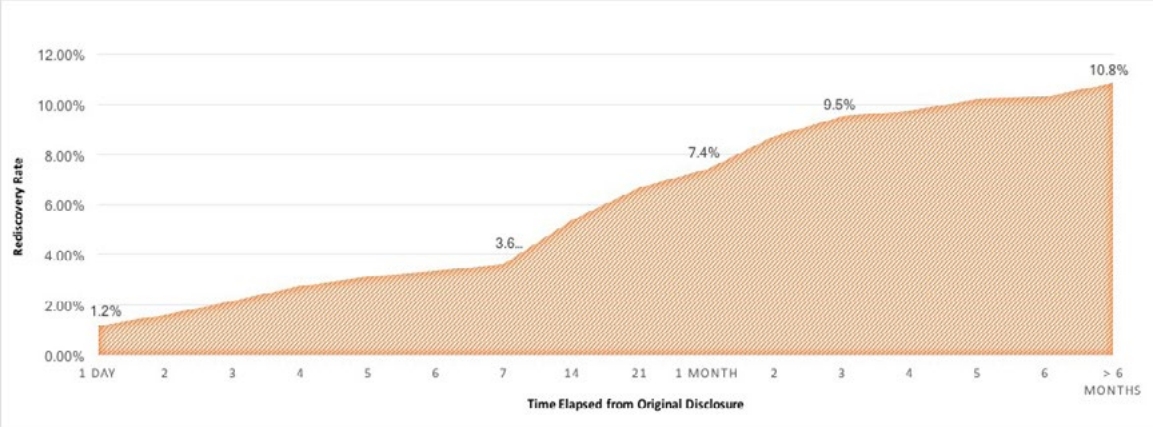
Within the first month after the original disclosure, more than 60% of vulnerabilities that will be rediscovered have been disclosed.[48]

Figure 8—Android Rediscovery Rate Over Time



Chrome is available with more granular date information. Figure 9 represents each day of the first week and the following three weeks as individual points on the x-axis before aggregating by month. As with Android, more than 60% of rediscovered vulnerabilities are disclosed within a month of the original bug report.[49]

Figure 9—Chrome Rediscovery Rate Over Time



48 This compares the aggregate rediscovery rate at one month (13.9%) to the aggregate one year Android rediscovery rate (21.9%). Within one month, 64% of rediscovery has taken place.

49 This compares the aggregate rediscovery rate at one month (6.9%) to the aggregate one year Chrome rediscovery rate (10.2%), yielding 68% of rediscovery.

5. Limitations of the Dataset

This research has to contend with several limitations. First, vulnerabilities are variably difficult to discover and exploit, which may influence discovery behavior and undermine the feasibility of aggregating them. Second, we only consider vulnerabilities with high or critical severity, which limits the generalizability of our results. There are additional factors over which we had no control that influence our result.

Failure to Report

The ability to count duplicate reports of a single vulnerability assumes people will continue to disclose their discoveries. This may be the case before the original vulnerability is made public, but will likely drop sharply afterward. This is generally because once a patch is available (even if not yet broadly applied), additional disclosures add little marginal value. For individuals with an interest in disclosure, the time to observe duplicates is during this rediscovery window—between the original vulnerability's disclosure and the developer making a patch available. Comparatively longer patch windows for some codebases could increase that time to capture duplicates, increasing the total number observed and skewing the rediscovery rate for that software.

Conversely, short rediscovery windows censor data that might otherwise be valid. One person affiliated with OpenSSL suggested that for open source projects, this patch window could be as short as seven days, leaving little chance for multiple parties to disclose.[50] The length of this rediscovery window is unlikely to impact criminal groups and others with much less interest in disclosing vulnerabilities. This leads to higher rates of rediscovery than are observed from only tracking disclosures. The format of disclosure behavior can also impact results; for example, differences may emerge between independent disclosure, bounty programs, and time-dependent events such as competitions and private bounty events. Discovery from an internal team may follow a different pattern altogether, as

50 Communication with the authors, 18OCT16.

recording a vulnerability allows employees to note the existence of the bug and pass it off to someone else.

Many of the same caveats that apply to Ozment's original analysis of Microsoft's vulnerability bulletins remain an issue here, "...the multiple individuals/organizations credited may have collaborated on finding the vulnerability, rather than identifying it independently. Furthermore, the... window of time for recording independent rediscoveries is [short]."[51] The dataset collected for this paper attempts to account for some of these concerns but is likely to lead to underestimates regardless.

Failure to Record

Vulnerabilities are also patched by internal quality assurance and security teams before they ever reach public versions of software. Individuals affiliated with companies such as Cisco and Apple, as well as ones associated with open source projects like OpenSSL, suggested that these internal fixes might go unrecorded. Where the public does find, and disclose, a vulnerability that has also been discovered internally, many groups are unlikely to record the subsequent disclosure as a duplicate. This failure to record vulnerabilities isn't limited to internal discovery. Groups sometimes differ on which bugs constitute vulnerabilities or fail to record additional disclosure after the original. OpenSSL provides a good case in point; we know from press reporting that two separate groups discovered the Heartbleed bug in April 2014, within days of each other, but only one group is credited for the disclosure in the OpenSSL records. There are likely to be additional cases of this sort this across vendors and open source projects.

There are also differences in the rediscovery rates of vulnerabilities at different severity levels, with some evidence that lower-severity flaws are rediscovered more often. Bugcrowd, a firm that helps integrate companies with a labor market for vulnerability discovery, found that rediscovery happened least often with their highest-severity bugs, 16.9% of the time. For second- and third-tier vulnerabilities (based on a five-tier system), the rediscovery rate jumped to 28.1% and 25.8%, respectively.[52] This sug-

51 Ozment, "The Likelihood of Vulnerability Rediscovery and the Social Utility of Vulnerability Hunting."

52 These figures were provided by Bugcrowd as an analysis of its internal reporting data, which integrates participants from both its public and private programs.

gests that rediscovery is more likely with less severe bugs—but this paper and the associated dataset are focused on the rediscovery rate and resulting policy implications from high- and critical-severity vulnerabilities. This is largely because of the potential consequences these vulnerabilities could have if used against computing systems and the Internet more broadly. The difference, at least in a narrow sample, between critical- and medium-severity vulnerability rediscovery rates does suggest that the estimates generated in this paper are lower than for the larger population of vulnerabilities.

Some factors contribute to over- rather than under-estimation. With bug bounty programs, disclosers will sometimes attempt to game the system by creating multiple aliases and submitting the same vulnerability repeatedly. In addition, at least one company's public bug-tracking system aggregated duplicates by linking separate vulnerability records and manually integrating them. This requires an analyst to discern the duplication and purposefully account for it. The degree to which bugs are duplicative can vary with individual judgment. Thus, our estimate of vulnerability records with two or more duplicates may be high.

The results of this analysis are significant as a baseline estimate representative of high- and critical-severity vulnerabilities. That the rediscovery rate for a broader set of software vulnerabilities may be higher only reinforces the utility of having a starting point to work from. The policy community doesn't address the issue of rediscovery in any way informed by empirics, and the scope of this dataset is far larger and more diverse than previous work. In addition, because of the emphasis on high- and critical-severity vulnerabilities, this paper prioritizes analysis of those bugs of greatest interest to the information security community.

Severity and Exploitability

Some of the bugs in this database, and thus a portion of our dataset of rediscoveries, may not be immediately exploitable by an individual. The paper doesn't assume anything about the individual discovering these vulnerabilities, including their skills or resources available to them to discover or operationalize any of these vulnerabilities. Indeed, one of the key

differentiators between this work and the RAND study as discussed in Section 6 is that the parties in our rediscovery could be any group rather than specifying a particular public/private combination.

This paper also uses the severity scores assigned to bugs by their respective databases, including Chromium, even though these may be inconsistent because of the behavior of ClusterFuzz and the degree of subjectivity in assigning severity of a bug. There are issues with the potential exploitability or severity of vulnerabilities in the Chromium dataset, but these are largely subjective judgments and people will continue to disagree with the original Chromium coding. To remove any bias on our end, we used the security severity ratings of the Chromium database, limiting the analysis only to high- and critical-severity vulnerabilities. While these may not all be *easily* exploitable, they all represent security holes in the browser.

Independence

It is not clear to what extent our bugs (or any in open source code) are truly independently discovered. There is a panoply of means by which vulnerability discovery could be made known to the world, from Twitter to new Metasploit modules to tracking disclosure credits. Following researchers like Tavis Ormandy and Natalie Silvanovich for several months prior to this publication, most observers would conclude there are vulnerabilities to be found in the Malware Protection Engine, and it could be a fruitful place to look for more. Disclosures stemming from this pattern of work by Ormandy and Silvanovich would thus not be strictly independent..[53] Managing this independence is an area of ongoing research but remains unresolved.

53 Iain Thompson, "'Crazy Bad' Bug in Microsoft's Windows Malware Scanner Can Be Used to Install Malware," *The Register*, May 9, 2017, https://www.theregister.co.uk/2017/05/09/microsoft_windows_defender_security_hole/.

Bounty Data

The data included in this research is not limited to vulnerabilities sourced from bounty programs. However, all the major vendors have bounties available for their software, though the reward structure and coverage varies over time and between programs. It is likely that the incentives offered by bounties bring more individuals into the pool to discover vulnerabilities in the corresponding codebase and changes the incentive to disclose. The same is almost certainly true of companies like Zerodium, which advertise multi-million dollar bounties for vulnerabilities in critical software like Apple's iOS.[54] The problem bounties, whether from vendors like Google or brokers like Zerodium, is that it remains unclear what systematic effect they have on disclosure behavior. There must be some effect, but absent a generalizable measure of how these bounties change discovery and disclosure incentives, no one can do more than acknowledge they are a factor.

6. Implications

This section deals with implications of this research and addresses the results of a similar and more recent study. There are three areas impacted by the findings from this study: how companies handle bug bounties, academic research into the malware markets, and government's disclosure of software vulnerabilities.

Patching from Bug Bounties

Bug bounties drive vulnerability disclosure to firms, and are a major way for companies to identify bugs to be patched. The volume of these bugs can be overwhelming, leading to prioritization of some patches to be completed more rapidly than others. Rediscovery rates can help drive that prioritization, pushing bugs with a higher rate of rediscovery toward the top of the patching queue. If two bugs of equal severity come from software with different rediscovery rates, the one more likely to be found multiple

54 Andy Greenberg, "Hackers Claim Million-Dollar Bounty for iOS Zero Day Attack," *WIRED*, November 2, 2015, http://www.wired.com/2015/11/hackers-claim-million-dollar-bounty-for-ios-attack/.

times should be patched first. This sort of awareness of how rediscovery patterns might inform new threats could bring useful new information to patch prioritization. This rediscovery information can also be useful to identify bugs that are more likely to appear many times in a bounty. Rediscovery rates of 10.8% to 21.9% in this paper can help set the parameters for what companies running bounty programs could expect—establishing a potential upper bound for vulnerabilities, especially those in open source software.

Studying the Malware Markets

Looking at the malware markets, rediscovery rates can help estimate product life cycles in malicious software. Higher rates of rediscovery will drive greater churn as exploit kits and other products dependent on vulnerabilities need to be refreshed more rapidly. At the rates of rediscovery found for this paper, nearly a fifth of all vulnerabilities may become known to a vendor or competitor in these markets every year. This discovery by other parties depresses the value of a vulnerability. These rates may also help inform a value curve for software vulnerabilities; as rediscovery rates go up, the period of high value shortens and absolute value may increase, owing to scarcity in these flaws.

The Vulnerability Equities Process

The VEP presents a trade-off between public security realized through updated and well-maintained software and that obtained by vigorous intelligence and law enforcement activity. In deciding to disclose a vulnerability, the government must weigh the relative costs of using a vulnerability and not disclosing it against turning that vulnerability over to a vendor so that it may be patched. If a vulnerability in the possession of the U.S. government is independently rediscovered by another party, the risk associated with keeping it secret is much higher than if it is never rediscovered.[55] This rediscovery rate becomes particularly important if the software in question

55 Assessing the entire chain of decision making in the mind of a notional attacker is impractical. Rediscovery is a critical point in the event chain. Without knowledge of a vulnerability, none of the other points about a potential attacker—motivation, skill, preferences—are relevant. In this scenario, a U.S. government agency is aware of the vulnerability and potentially has it in active use, leading to a decision not to disclose the information to the affected software's vendor. While in use, this vulnerability is then discovered by a malicious party. This rediscovery means the vulnerability information is in several people's hands and no longer exclusive to the U.S.

is a widely used open source project or a cryptographic library. Here, the ripple effects of non-disclosure could impact a far larger number of people.

What do the rediscovery rates from this paper tell us about the VEP? There is little systematic transparency into the types or number of vulnerabilities held by the U.S. government. It would be helpful if we could start to narrow down what might make intelligence community vulnerabilities distinct from all others, perhaps by some combination of measures for depth in codebase, discoverability, exploitability, and usefulness once exploited. The evaluation of what makes a vulnerability worthy of intelligence community interest appears to be a mix of opinion and anecdotal evidence. This is insufficiently systematic to make strong claims about the population of intelligence community vulnerabilities.

What it means for this paper is that we cannot make a strong claim about what effect our rates of rediscovery might have with respect an intelligence agency's stock of vulnerabilities. Throughout this paper, we have been careful to clarify that our findings applied to the data available to our analysis and shouldn't be taken to apply to all bugs across time. In a previous version of this paper, as a thought experiment to demonstrate the significance of a few percentage points difference in rediscovery rate, we applied our numbers to an estimate of the total NSA stock of vulnerabilities generated last year by a group of researchers at Columbia University. [56]

The challenge with this discussing vulnerabilities retained by the U.S. government that no research available in the public domain has yet demonstrated systematically what the character and distinctive nature of government held vulnerabilities might be. Thus, any approximation can only be an educated guess. While there are critical security issues among the vulnerabilities contained in this database, including exactly the sort of sandbox escapes and memory corruption errors that one presumes an intelligence agency might be interested in, we can not be certain.

What we can say is that our rediscovery rate data indicates vulnerabilities are found more often than previously thought, between 10.8% and 21.9%

56 Healey, "The U.S. Government and Zero-Day Vulnerabilities: From Pre-Heartbleed to Shadow Brokers."

of the time within a year. Rediscovery captures only part of the population of vulnerabilities held secret by the U.S. government but released into the wild with little to no warning. There is also proliferation of these tools through theft; take as example the Shadow Brokers release of the Equation Group tools and continuing disclosure of NSA vulnerabilities, including the one that made the WannaCry worm possible.[57] These findings, together with legitimate concerns over the management of risk associated with U.S. cybersecurity capabilities, should drive policymakers to reevaluate the current standards for government disclosure of software vulnerabilities. These findings should also motivate critical discussion and support for new research, preferably unclassified, about how these rediscovery rates differ for high-consequence software such as common cryptographic libraries or frequently used embedded systems software components.

Analysis in the Context of Recent Work

The RAND Corporation published a study on zero-day vulnerabilities a few days after this paper began circulating for comment. Though dealing with a larger range of issues in the development and use of vulnerabilities, the RAND study also addressed bug collisions, a topic similar to vulnerability rediscovery, which has led many to compare results from the two papers.[58] This paper finds that 10.8% to 21.9% of vulnerabilities are rediscovered within a year, but the RAND study finds the rate to be 5.8%. Though the two papers are asking slightly different questions, as explained below, we disagree with their conclusion that the rediscovery rate is as low as 6% over a year and especially that it remains below 1% within 90 days of initial discovery. Figure 10 shows our data on the rediscovery rate over time for both Android and Chrome, overlaid on each other. Much of Chrome's rediscovery takes place before the end of the first month, rising slowly to 9.5% at three months. Android spikes rapidly from 5.7% in less than a month, through 13.9% at one month, to 20.2% at three months.

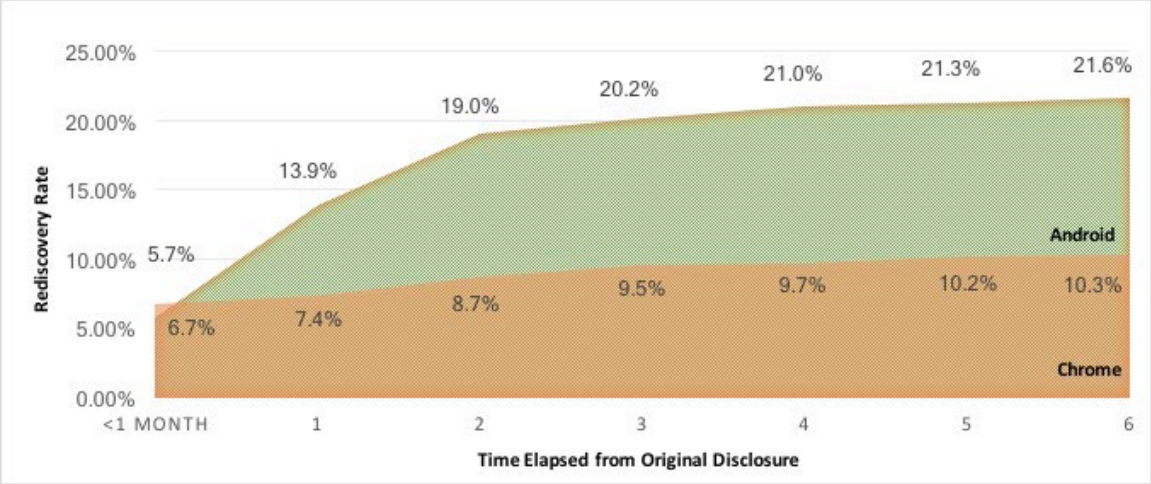
---

57 Bruce Schneier, "Major NSA/Equation Group Leak - Schneier on Security," *Schneier on Security*, August 16, 2016, https://www.schneier.com/blog/archives/2016/08/major_nsaequati.html.3,23]]}}}],"schema":"https://github.com/citation-style-language/schema/raw/master/csl-citation.json"} ; For more on intentional vs. unintentional proliferation, see Trey Herr, "Governing Proliferation in Cybersecurity," *Global Summitry* 2, no. 1 (July 2017).

58 Snowden, Edward. Twitter post. March 11th, 2017. https://twitter.com/Snowden/status/840602409734922241 & Kim Zetter, "Malware Attacks Used by the U.S. Government Retain Potency for Many Years, New Evidence Indicates," *The Intercept*, March 10, 2017, https://theintercept.com/2017/03/10/government-zero-days-7-years/.

Both rates are substantially higher than the 90-day rate from the RAND study. While some of this difference can be accounted for by the different sources of data and the two studies posing somewhat different questions, the gap remains striking.

Figure 10—Android and Chrome Rediscovery Rates over Time



The discrepancy between these two studies largely stems from differences in their questions and methodology. The RAND team worked with a vulnerability research group to construct a sample matching what might be found in the intelligence or military community. From their study: "We believe these data are relatively representative of what a sophisticated nation-state might have in its arsenal...applying wide generalizations to other datasets may be misleading, as generalizations to other data can only be drawn if the data are similar in nature to ours."[59] This issue in data source is not an inconsiderable one; vulnerabilities discovered and reported directly to a vendor (whether through a bounty or not) will cover a wider array of bug types and severity than those selected strictly for operational use. It's not clear to what extent this overlap is addressed by our sample being constrained to high- and critical-severity vulnerabilities.

There are other differences, rooted in the data used by each study. RAND's total dataset constitutes 207 vulnerabilities spanning 14 years, and accurate information on dates such as vulnerability birth and maturity were only available for 61% of these bugs, thus potentially reducing the useful dataset size for this rediscovery question to approximately 127 vulnerabilities, or

59 Ablon and Bogart, "Zero Days, Thousands of Nights," 61.

9 per year.[60] Our study examined 2,6709 vulnerabilities, spanning 9 years, from open source software, including Chrome and the Android operating system. Every vulnerability record used in our analysis, with a minimum of their corresponding CVE or bug number, severity score, and dates for disclosure and duplicates, are available in a GitHub repository, along with all the scripts used to extract, clean, and organize this data.[61] While the RAND dataset hasn't been made public, summary statistics from the paper indicate approximately 60% of the vulnerabilities affect closed source systems, predominantly Microsoft products, while another 35% are open source, and 5% of undeclared origin.[62]

One of the key dimensions of the vulnerability disclosure debate is the frequency with which a flaw held secret by the U.S. government will be discovered by another party; both studies address this issue but each does so through a different question. This paper looks at independent rediscovery of vulnerabilities: instances where two independent parties disclose the same vulnerability to a vendor. This data represents high and critical vulnerabilities and measures the instance of independent discovery of the same bug between two parties, without characterizing the skill or resources of the discovering party.

RAND looked at the frequency with which vulnerabilities in the public domain collided with previously known vulnerabilities in a private dataset. The team's key claim is that this dataset more closely represents what an intelligence agency might use, thus measuring the chance for two *unequal* parties to find the same vulnerability: a government and researchers/criminals. Given the lack of a systematic model for how the capability (and capacity) to discover vulnerabilities is distributed across the malware markets, including state organizations and companies, we should hesitate to make strong claims about how one group might represent another. [63]

60 Ibid., 15.

61 Christopher Morris, Trey Herr, and Amy Armbrust, "Vulnerability-Rediscovery," GitHub, *MASE*, (2017), https://github.com/mase-gh/Vulnerability-Rediscovery.

62 Ablon and Bogart, "Zero Days, Thousands of Nights," 101.

63 Jaziar Radianti and Jose J. Gonzalez, "Dynamic Modeling of the Cyber Security Threat Problem: The Black Market for Vulnerabilities," *Cyber-Security and Global Information Assurance: Threat Analysis And Response Solutions*, 2009, http://www.igi-global.com/chapter/dynamic-modeling-cyber-security-threat/7408; Jaziar Radianti, "Eliciting Information on the Vulnerability Black Market from Interviews" (Fourth International Conference on Emerging Security Information, Systems and Technologies, IEEE, 2010), 93–96, doi:10.1109/SECURWARE.2010.23; Mingyi Zhao, Jens Grossklags, and Peng Liu, "An Empirical Study of Web Vulnerability Discovery Ecosystems," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (ACM, 2015), 1105–1117, http://dl.acm.org/citation.cfm?id=2813704.

Vulnerability Types

It is difficult to say with precision what systematic differences exist between vulnerabilities collected by the intelligence community and our dataset. Accurate estimation of rediscovery matters for a range of topics, including the VEP, research on the malware markets, and analyzing the efficacy of bug bounty programs. In looking at VEP, there is reason to believe that vulnerabilities collected and employed by the intelligence community are not a simple cross-section of all vulnerabilities. Software flaws in use by the intelligence community are likely to skew toward those most useful for gaining and maintaining access to the computer systems of intelligence collection targets. They may also be influenced by the individuals and techniques used in discovering vulnerabilities. A common refrain is that the NSA places a significant emphasis on mathematical capabilities in vulnerability discovery. This emphasis could result in NSA's collection of vulnerabilities containing far more cryptographic flaws than would be found in a random sample of software vulnerabilities anywhere in the public domain. The same is likely true to some degree of law enforcement organizations.

The vulnerabilities examined in this paper are from open source software, including Chromium and the Android operating system, which means they represent a particular subset of the total population of software. Vulnerabilities in both systems would be useful to gain access to protected information such as in Chrome, where our sample includes several sandbox escapes.[64] Our analysis assumes nothing about the parties involved in rediscovery except that they are working independently. Because of this, we cannot characterize rediscovery as more or less likely to impact the vulnerabilities held by an intelligence or law enforcement organization.

There is very little data on vulnerability rediscovery or collision, however framed, and so we must be careful about data and our methodologies. The RAND team sourced its data from a private organization, and some of the vulnerabilities in that dataset may have been in use or for sale, but the study disclosed little more than summary statistics about its data. The small size of the dataset, as well as its continued secrecy, makes effective

64 Includes CVE-2014-5332 and 2016-1706

peer-review or replication very difficult. That said, the RAND paper is additive, including substantial material on the life and times of zero-day vulnerabilities, which will benefit policymakers and scholars alike.

7. Conclusions

This paper provides the first transparent empirical study of vulnerability rediscovery in multiple types of software and across different vendors, considering the rate of rediscovery, the impact of time, the length of lag between original and duplicate discoveries, and the variation of each of these factors across different vendors. Where previous work has estimated rediscovery rates for software between 5% and 9% largely unbounded by time, this paper demonstrates that rates for high- and critical-severity bugs are higher, between 10.8% and 21.9% across the software surveyed.

The impact of rediscovery scales with the number of vulnerabilities in a codebase. Looking at Android, for example—in 2016 there were 287 recorded high- and critical-severity vulnerabilities. At a rediscovery rate of 6%, closer to Ozment's original estimates, Google should expect that at least 17 vulnerabilities known to them in 2016 will have been discovered by another party, potentially before being disclosed. Using the 21.9% rate found above, that number jumps to 63 vulnerabilities.

At that rate, nearly 50 additional software vulnerabilities are discovered each year above previous estimates, many of which could be used by criminal groups or states before the developer patches them. The presence of these vulnerabilities in the malware markets means they may also be integrated into other malicious tools and see their lifespans extended even further. For the scholarly community, the current state of practice in tracking vulnerability disclosures and rediscovery is poor. This undermines the community's ability to track the longer-term efficacy of investment in

secure development practices and bug bounty programs.[65] It is also relevant to the discussion that even when there is a patch available, many vulnerabilities don't immediately become useless, as users and organizations often delay applying patches for months or longer.[66] Thus, a patched vulnerability still can still have operational utility.

The relatively long lag time between original disclosure and rediscovery, more than two months on average, based on data from the Android operating system, suggests that many rediscoveries are truly independent, thus providing a more accurate model of the behavior of malicious third parties. When the rediscovery window is short or nearly zero, it suggests that discoverers communicated about, or were aware of, each other's efforts. This would inflate the rediscovery rate but hide the fact that many were working from the same information. There are many reasons to believe that the rediscovery rates presented in this paper are an *under*estimate of the true rate of rediscovery including records from Bugcrowd which indicate that low- and medium-severity vulnerabilities are rediscovered more frequently than the high- and critical-severity bugs to which this study is constrained.

These findings do not consider classified data. It is possible that the intelligence, defense, or law enforcement communities have conducted studies of rediscovery and reached different conclusions; however, there is no evidence of this in the public domain. Our estimates are based on data available to all researchers, and evaluate software in common use by U.S. citizens and many government employees. While secret studies may exist,

65 Chad Heitzenrater, Rainer Böhme, and Andrew Simpson, "The Days before Zero Day: Investment Models for Secure Software Engineering," in *Proceedings of the 15th Workshop on the Economics of Information Security (WEIS)*, 2016, http://weis2016.econinfosec.org/wp-content/uploads/sites/2/2016/05/WEIS_2016_paper_21-2.pdf; Pontus Johnson et al., "Time between Vulnerability Disclosures: A Measure of Software Product Vulnerability," *Computers & Security* 62 (2016): 278–295.

66 Terry Ramos, "The Laws of Vulnerabilities," in *RSA Conference*, 2006, http://www.qualys.de/docs/Laws-Presentation.pdf; Sandy Clark et al., "Moving Targets: Security and Rapid-Release in Firefox," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (ACM, 2014), 1256–1266, http://dl.acm.org/citation.cfm?id=2660320; Zakir Durumeric et al., "The Matter of Heartbleed," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14 (New York, NY, USA: ACM, 2014), 475–488, http://dl.acm.org/citation.cfm?id=2663755; Antonio Nappa et al., "The Attack of the Clones: A Study of the Impact of Shared Code on Vulnerability Patching," in *Security and Privacy (SP), 2015 IEEE Symposium on* (IEEE, 2015), 692–708, http://ieeexplore.ieee.org/abstract/document/7163055/; Armin Sarabi et al., "Patch Me If You Can: A Study on the Effects of Individual User Behavior on the End-Host Vulnerability State," in *International Conference on Passive and Active Network Measurement* (Springer, 2017), 113–125, http://link.springer.com/chapter/10.1007/978-3-319-54328-4_9.\\uc0\\u8220{}The Matter of Heartbleed,\\uc0\\u8221{} in {\\i{}Proceedings of the 2014 Conference on Internet Measurement Conference}, IMC \\uc0\\u8217{}14 (New York, NY, USA: ACM, 2014

the data in this paper includes software that would have to be part of any such study. Our estimates are higher than previously reported in scholarship available to the public, and these findings should speak directly to the policy community, regardless of classification.

The Cyber Security Project
Belfer Center for Science and International Affairs
Harvard Kennedy School
79 John F. Kennedy Street
Cambridge, MA 02138

www.belfercenter.org/Cyber

Printed in the United States of America