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
JULY 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
JULY 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 acknowledges support from the Belfer Family and the Flora and William Hewlett Foundation. The 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, Tod Beardsley, 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 discovered 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 4,300 vulnerabilities, and estimates vulnerability rediscovery across different vendors and software types. It concludes that rediscovery happens more than twice as often as the 1-9% range previously reported. For our dataset, 15% to 20% of vulnerabilities are discovered independently at least twice within a year. For just Android, 13.9% of vulnerabilities are rediscovered within 60 days, rising to 20% within 90 days, and above 21% within 120 days. For the Chrome browser we found 12.57% rediscovery within 60 days; and the aggregate rate for our entire dataset generally rises over the eight-year span, topping out at 19.6% in 2016. We believe that the actual rate is even higher for certain types of software.

When combined with an estimate of the total count of vulnerabilities in use by the NSA, these rates suggest that rediscovery of vulnerabilities kept secret by the U.S. government may be the source of up to one-third of all zero-day vulnerabilities detected in use each year. These results indicate that the information security community needs to map the impact of rediscovery on the efficacy of bug bounty programs and policymakers should more rigorously evaluate the costs of 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 *re*discovery. 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 both 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, our research, and addresses some of the larger questions raised by this scholarly collision. A particularly challenging issue with rediscovery is that this rate changes over time and varies for different types of software. We collected data from multiple software vendors and an open source project 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 use 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.

# 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. Such 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.[2] Vulnerabilities may also be introduced directly to hardware through

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

compromises in chip design or manufacture somewhere along the supply chain.[3]

Not all vulnerabilities are created equal—some are easier to find than others and only a small number will provide easy 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 which can immediately be used to gain access to a computer system to develop a useful exploit.

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.[4] 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.[5]

3 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/.

4 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.

5 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*.

## Value of Rediscovery

Because vulnerabilities are non-rivalrous, they can be discovered and held by more than one party simultaneously. A rediscovered vulnerability may be the same as the original, or a closely related flaw deemed similar enough to be 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 which had previously only been known to a single party enters the public domain. Asking about the rate of rediscovery only assumes that the two groups that find the same bug are independent of each other.

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.[6] The key choice for this VEP process is whether to keep the vulnerability secret or disclose the flaw to its developer, whereupon the government loses the opportunity to use the vulnerability (though the time this takes to happen can vary dramatically). Measuring the cost of disclosure vs. 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.[7] 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.

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

6 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.

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

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 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 vs. salted herring, some vulnerabilities are likely to "go stale," and thus be of little value, much faster than others.[8] 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.[9] 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.[10] This paper also

8 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.

9 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.

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

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.

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.[11]

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.[12] 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 there was little long-term utility to vulnerability discovery and patching.[13] Finifter and his group were studying the efficacy of bounties over 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

11 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.

12 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.

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

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 vulnerabilities—those used by attackers before the vendors learn about them.[14] 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 much lower than our findings that 13% to 20% of vulnerabilities are rediscovered within a year, including more than 21% in the Android operating system. 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.

14 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.

# 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.[15] 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.[16]

## 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 disclosure. This 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.

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

16 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.

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? We employ one or both approaches with each codebase. More detail on the specific method of counting duplicates for each piece of software can be found below.

Calculating rediscovery, we take measure of the number of all vulnerability records with duplicates as a proportion of all vulnerability records. 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 1/10 or 10%.[17] In our estimates, we collect the total population of vulnerabilities and sample those of high or critical severity to measure this rediscovery rate.

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 grow in the months after a vulnerability's initial disclosure, eventually leveling off within a few months. Rediscovery over time measures this rate of change, the brevity of which suggests that the distribution of discovery attention is not uniform and changes over time. 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 DuplicateA would be **X** while the lag for DuplicateB would be **X** + **Y**.

17 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.

## 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. 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 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."[18]

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.

Data for this range of products and vendors 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

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

limiting our collection to these high- and critical-severity vulnerabilities, the dataset is made less generalizable but more reliable.

- Firefox: The Firefox dataset is scraped from records in the Bugzilla bug tracker for Firefox and related software dependencies.[19] This data constitutes all bugs labeled as high-priority or critical in the Severity field from Extended Service Release (ESR) advisories for Firefox between 2012 and 2016 and comprises **473 vulnerability records** and **81 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.

- Chrome: The Chrome dataset is scraped from bugs collected for Chromium, an open source software project whose code constitutes most the Chrome browser.[20] On top of Chromium, Google adds a few additional features, such as a PDF viewer, but there is substantial overlap, so we treat this as essentially identical to Chrome.[21] Chrome presented a similar problem to Firefox, so to

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

20 The Chromium Projects, "Chromium Bug Tracker," *Bugs*, 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.

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

record only vulnerabilities with a reasonable likelihood of public discovery, we limited our collection to bugs labeled as high or critical severity from the Chromium bug tracker. This portion of the dataset comprises **3,397 vulnerability records** of which there are **468 records with duplicates**. For Chrome, we coded a vulnerability record as a duplicate if it had been merged with another, where merges were noted in the comments associated with each vulnerability record, or marked as a Duplicate in the Status field.

- OpenSSL: Our data comprises all vulnerability records from 2014 to 2016 for the OpenSSL project.[22] The start time for this window is the disclosure of Heartbleed. Based on conversations with members of the OpenSSL community, the consistency of record keeping appears to have improved commensurate with a new influx of resources after the Heartbleed disclosure in April 2014. This portion of the dataset comprises **85 vulnerability records** of which **2 have duplicate disclosures.**[23] 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 so not duplicate disclosures. Records with an '&' between reporters are coded as independent discoveries and thus duplicates.

- Android: We were provided access to data from Google's tracking systems for Android vulnerabilities: the public system, CodeSite, and an internal-use-only platform called Issue Tracker. The internal site track used by Google employees tracks disclosures from within the company as well as some from outside while CodeSite records disclosures from the public. 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 the Google security team downloaded bugs from both tracking systems, then correlated them to remove identical records that existed in both systems.[24] This individual then added a CVE ID

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

23 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.

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

for all records, to replace the internal tracking IDs. Instances of rediscovery, duplicate disclosures, were coded when a vulnerability record in CodeSite was noted as a duplicate and linked to a report credited to a separate researcher(s) from the original. Duplicates were also coded if two bugs were merged together. The Android portion of the paper's dataset comprises **352 vulnerability records** where **77 records had at least 1 duplicate**.

**Table 1—Dataset Summary**[25]

| Source | Date Range | Total Population | Sample Vulnerabilities | Sample Duplicates | Rediscovery Rate |
|---|---|---|---|---|---|
| *Google—Chrome* | 2009–2016 | 6817 | 3397 | 468 | 13.8% |
| *Mozilla—Firefox* | 2012–2016 | 1112 | 473 | 81 | 17.1% |
| *Google—Android* | 2015–2016 | 682* | 352 | 77 | 21.9% |
| *OpenSSL* | 2014–2016 | 85 | 85 | 2 | 2.4% |
| ***Total*** | **2009–2016** | **8696** | **4307** | **628** | **14.6%** |

The above table summarizes the four sources for this paper's dataset, covering more than 4,300 vulnerability records over eight 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 available from each of the four data sources, of all criticality levels, from the date range specified.
- Sample Vulnerabilities—our sample of high and critical vulnerabilities, a subset of the total population of vulnerabilities for the source software

25 * 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

- 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.

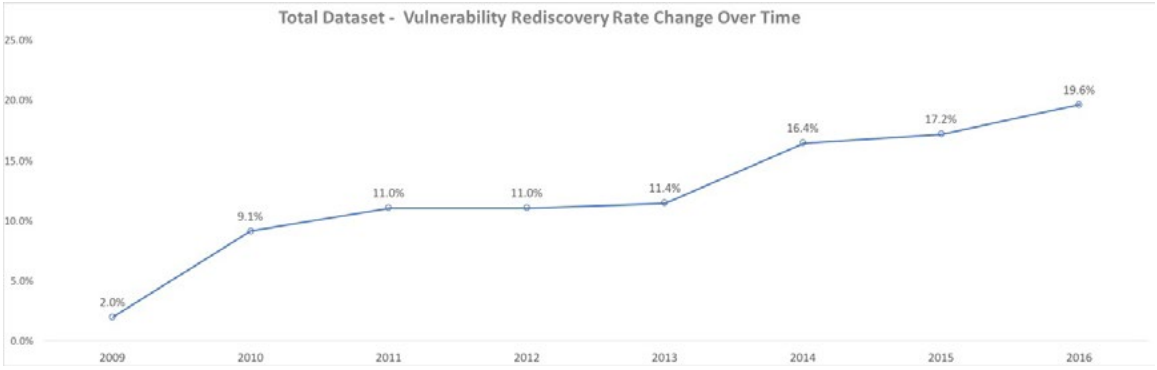
# 4 Analysis

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 eight-year span of this dataset see an aggregate 14.9% rate of rediscovery. This is higher than previous open-source estimates, which ranged from 6.84% in early empirical work to 9% in more recent simulations. **Figure 1** below charts the annualized discovery rate over the whole dataset.

**Figure 1—Aggregate Vulnerability Rediscovery Rate Change Over Time**




This sample of high and critical vulnerability records shows a rate of rediscovery that is twice as high as previously thought, and which has been steadily increasing over the past decade in the software we examined.

## 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. The three charts below in **Figures 2–4** 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
- 1 Duplicate: vulnerabilities with an initial disclosure and one duplicate
- 2 or more Duplicates: vulnerabilities with an initial disclosure and more than one duplicate. Note that this

counts all vulnerability records with more than 1 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 duplicate and those with two or more.

**Figure 2—Firefox Vulnerability Rediscovery**

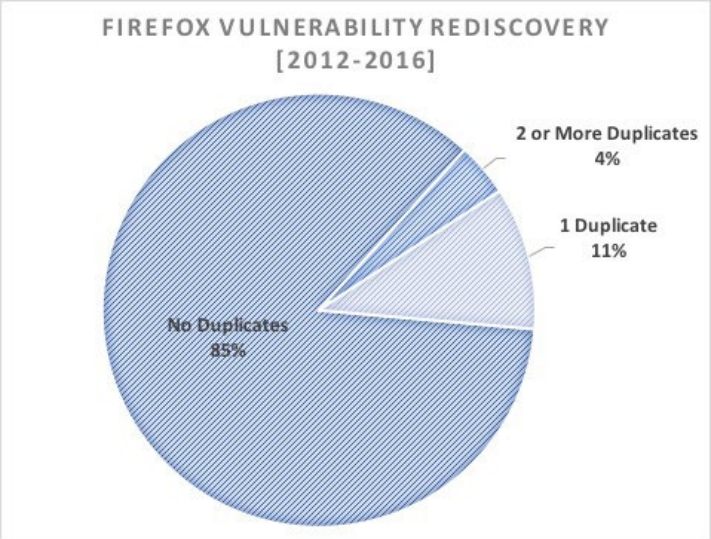



**Figure 3– Chrome Vulnerability Rediscovery**

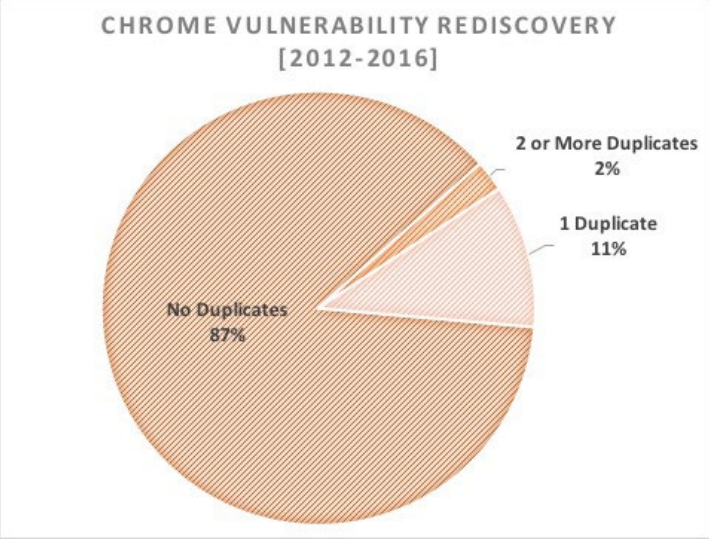
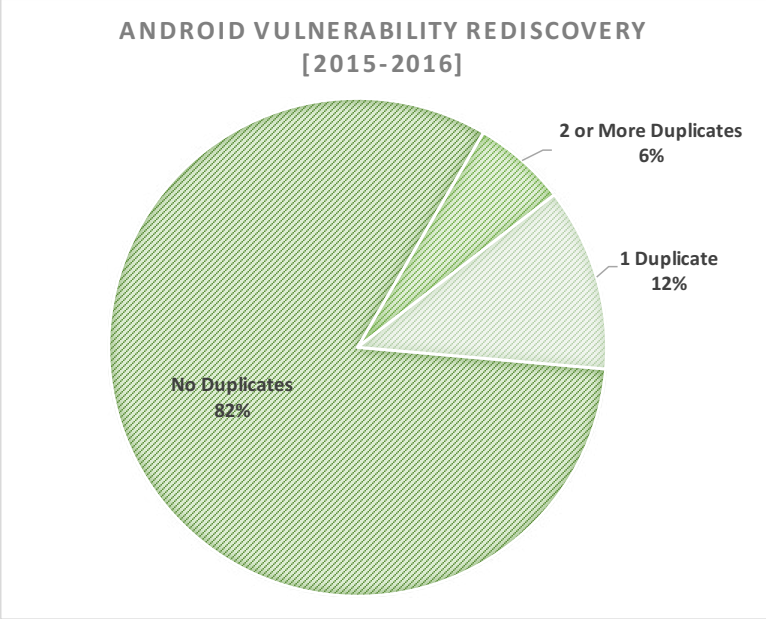


**Figure 4—Android Vulnerability Rediscovery**




Looking at the three codebases, there is a trend where small increases in the overall rediscovery rate are largely accounted for by vulnerabilities with two or more instances of rediscovery. That is, newly rediscovered vulnerabilities appear to be shallow and more likely to be found by multiple parties.

## Vulnerability Rediscovery Lag

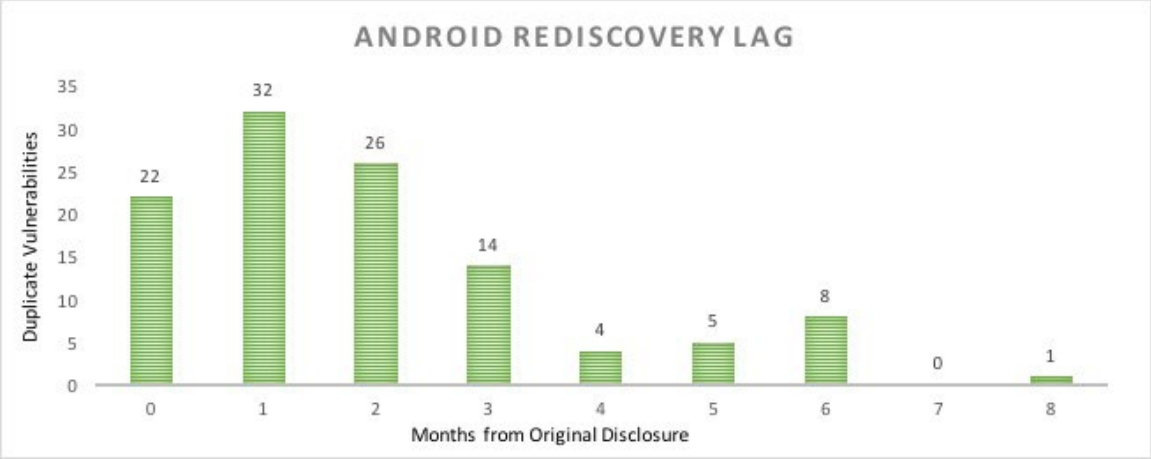
Rediscovery can be a useful metric, but we should also consider the element of time and its impact on subsequent additional discoveries of the same vulnerability. This section looks at the time between original and duplicate disclosure, "the rediscovery lag." Where that lag is longer, it suggests discoverers are stumbling across the same bug with something approximating truly independent discovery. This section uses the Chrome and Android data both to highlight differences between software types and to make use of the more complete timing data available for both.

Rediscovery lag was calculated by taking the absolute time difference between original vulnerability disclosure and subsequent duplicates. For

additional duplicates after the first, the time lag was measured between their disclosure date and that of the original. The Android data for this paper was made available with assistance from Google and comes from a combination of public and internal bug tracking systems. This section presents the Android data sorted into discrete monthly categories because of the inability to collect original and duplicate disclosure dates for each vulnerability record. So, if a vulnerability is discovered at $t_0$, rediscovered once at $t_{30}$, and then rediscovered again at $t_{60}$, the chart would include both disclosures—one each in the one- and two-month categories.

In looking at Android, the average time between initial discovery and rediscovery was two months, though a sizable number also occurred within the same month as the original disclosure (0 months). **Figure 5** shows a distribution of Android rediscovery lag for all duplicates, including vulnerabilities with multiple duplicates, covering 112 duplicates for 77 distinct vulnerabilities.

**Figure 5—Android Rediscovery Lag**



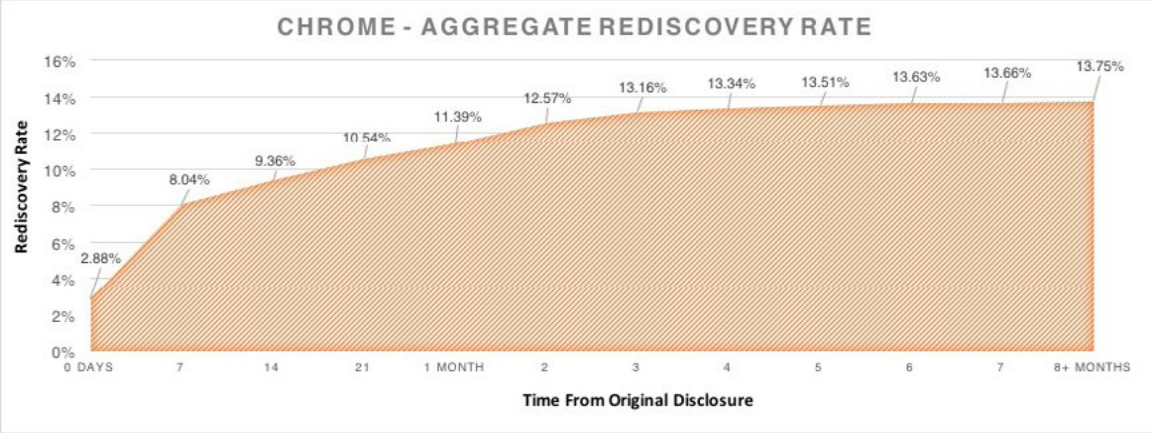

The disclosure of most duplicate vulnerabilities in Android takes place months later than the original, while only 20% occur in the same month. This pushes back on the notion of rediscovery as being largely "simultaneous discovery" and suggests instead a more independent activity.

Figure 6 shows the time from original to first rediscovery for a subset of the vulnerabilities in Chrome. There is a wider distribution, including multiple vulnerabilities spread all the way to eight months, but a strong concentration in the first seven days after the original. This may suggest greater communication within the Chrome security community either directly or through more transparency in the Chromium bug tracker.

## 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 is what proportion are likely to be rediscovered within a given time? This section looks at data for Android and Chrome to show the same differences between codebase as in rediscovery lag. With Chrome, nearly 40% of vulnerabilities are reported within the same 24-hour period, rising quickly to the Chrome lifetime average of about 14% rediscovery, as shown in **Figure 7**.

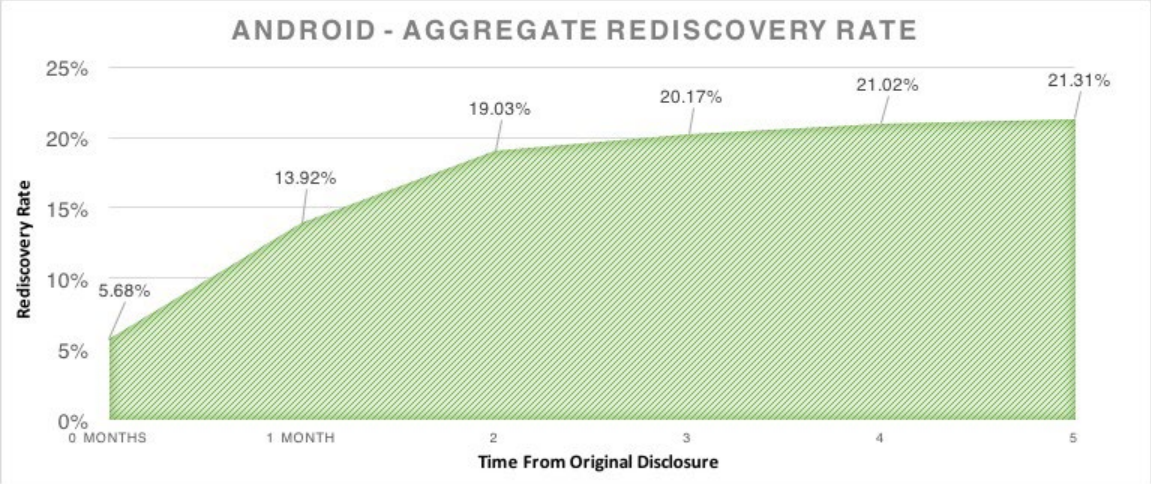
**Figure 7—Chrome Aggregate Rediscovery Over Time**




For Android, that rediscovery is slower to increase but ends up higher. While the Android data is sorted into months and so does not cover the week-to-week detail in the first month as does the Chrome data, there is a nearly identical rate of same-day rediscovery, while the following two

months take more time to reach the near peak rate at 19%. Rediscovery rates rise quickly in the first 90 days, then largely level off as shown below in **Figure 8**.

**Figure 8—Android Aggregate Rediscovery Rate**




For a more direct comparison of these two, look to **Figure 9** (page 28), where data from Chrome is sorted into months in a similar fashion as Android and the rediscovery over time charts for both are overlaid.

# 5 Limitations of the Dataset

This research deals 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 that influence our result over which we had no control.

*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.[26] 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 just from 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 recording a vulnerability allows employees to note the existence of the bug and pass it off to someone else.

26 Author Communication, 18 Oct 2016.

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]."[27] 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 fixed by internal quality assurance and security teams before they ever reach public versions of software or duplicates not recorded after the original disclosure. In talking to individuals affiliated with companies like Cisco and Apple, as well as open source projects like OpenSSL, many suggested 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 disclosure in the OpenSSL records. There are likely additional cases like 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.[28] This suggests 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 consequence these vulnerabilities

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

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

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.

# 6 Implications

This section deals with implications from this research and addresses the results of a similar and more recent study. There are three principle implications for the rediscovery rates we find in this paper, looking at how companies handle bug bounties, academic research into the malware markets, and for government's handling of bugs in the vulnerability equities process (VEP).

*Patching from Bug Bounties*

Bug bounties drive vulnerability disclosure to firms, 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 chance of being discovered by other parties while in the patching process towards the top of the list. Rediscovery rates of 15 to 20% in this paper 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 (VEP)*

The VEP presents a balancing issue, 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.[29] This rediscovery rate becomes particularly important if the software in question 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 VEP? There is little systematic transparency into the types or number of vulnerabilities held by the U.S. government, but by combining our work with a report produced by Columbia University, which estimates the U.S. government retains no more than 50 to 250 vulnerabilities for use at any given time, we can offer some answers.[30] Using the aggregate rediscovery rate for our entire dataset, 14.9%, means that in a given year, somewhere between 8 and 37 vulnerabilities kept secret by the U.S. will be rediscovered by other parties.

Not every one of these vulnerabilities, in active use by law enforcement and the intelligence community, may be rediscovered, or could be capably employed, by potential adversaries. Making a conservative assumption, that only 50% of these rediscovered flaws are both useful and in the hands of malicious actors, a 15% rediscovery rate still leaves 4 to 18 high value software flaws in the wild every year, and likely to be discovered by attackers before software vendors, potentially a major source of new zero-days. In 2016, Symantec reported that there had been 54 zero-day vulnerabilities

29 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.

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

used in 2015.[31] This means that the rediscovery of vulnerabilities kept secret by the U.S. government for operational use could contribute anywhere from 7.5% to 33% of this 2015 zero-day population.

Without accounting for the unintentional proliferation of new vulnerabilities from leaks like the Equation Group tools released by Shadow Brokers, this proportion of zero-days potentially left unpatched and exploited while their underlying vulnerabilities are in use by the U.S. government is significant.[32]

These findings suggest that policymakers should reevaluate current practice in use and retention of vulnerabilities, that up to a 1/3 of all zero-days in use by other states and criminal groups may come from U.S. failure to disclose vulnerabilities to vendors changes the state of the debate. These findings should also motivate critical discussion on the standards of vulnerability disclosure review and prompt support for new research, preferably unclassified, about how these rediscovery rates differ for high-consequence software such as common cryptographic or common software libraries for embedded systems.

*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 similar topic to vulnerability rediscovery, which has led many to compare results from the two.[33] This paper finds that 15% to 20% of vulnerabilities are rediscovered within a year, but the RAND study finds the rate to be about 6%. Though the two papers are asking slightly different questions, as explained below,

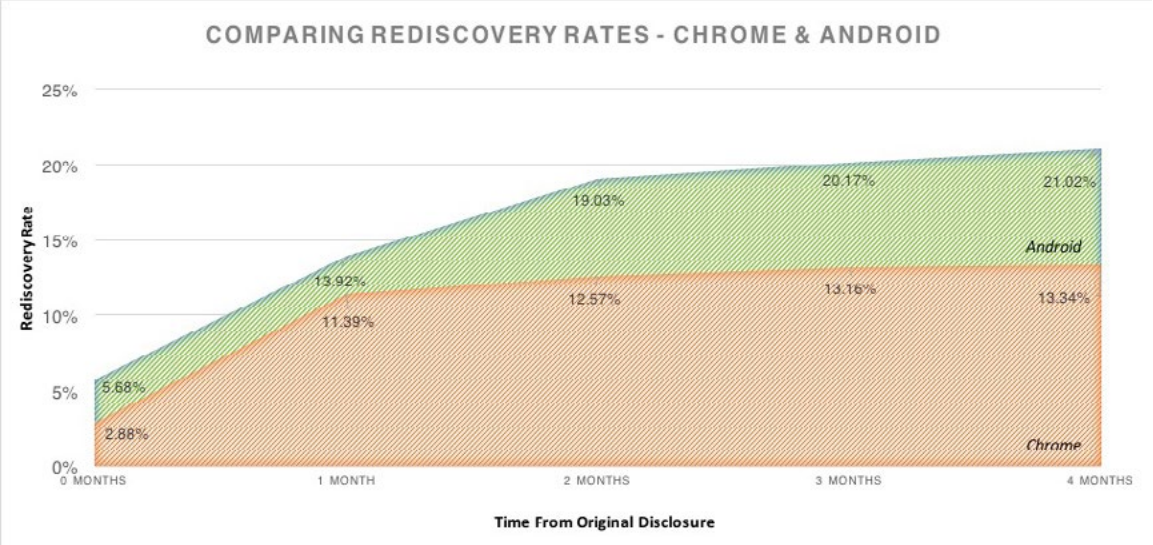
31 Symantec, "Internet Security Threat Report," ISTR (Symantec, April 2016), https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf.

32 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 (June 2017).

33 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/.

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 9 shows our data on the rediscovery rate over time for both Android and Chrome, overlaid on each other. While the discovery rate at a month or less hovers between 2% and 5%, the rate rises quickly for both codebases—rising to just above 13% in Chrome and above 21% in Android.[34]

**Figure 9—Android and Chrome Rediscovery Rates over Time**




The disagreement 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."[35] 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.

34 The Chrome data has been sorted in the same way as Android for purposes of comparison here.

35 Ibid., 61.

There are other differences, rooted in the data used by each study. RAND's total dataset constitutes only 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 9 per year.[36] Our study includes 4,307 vulnerabilities, spanning 8 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.[37] 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.[38]

In terms of the VEP, a key question is this: how often is a vulnerability held in secret by the U.S. government going to be found by another party, whether state or non-state? To answer this, the two studies end up asking slightly different questions of their data. 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 making any claim about their capability to discover or use a vulnerability. RAND looked at the frequency with which vulnerabilities in the public domain collided with previously known vulnerabilities in a private dataset. Their 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

36 Ibid., 15.

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

38 Ibid., 101.

companies, we should hesitate to make strong claims about how one group might represent another. [39]

It is difficult to say with precision what systematic differences exist between vulnerabilities collected by the intelligence community and our dataset. Estimating 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 towards 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 discovery vulnerabilities. A common refrain is that the National Security Agency (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 only a portion 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.[40] Our analysis does not presume that we know either party involved in rediscovery other than that they are working independently. Because of this, we cannot characterize that rediscovery is more or less likely to impact the vulnerabilities held by an intelligence or law enforcement organization.

39 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.

40 Includes CVE-2016-1706 and 2014-5332

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 the dataset may have been in use or for sale, but the study disclosed little more than summary statistics about their data. The small size of the dataset as well as its continued secrecy makes effective 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 empirical study of vulnerability rediscovery in multiple types of software and across different vendors, considering the rate of discovery, 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, as much as 23% within a year for Android. Table 2 summarizes annual rediscovery rates for Chrome, Firefox, and Android.

**Table 2—Rediscovery Rates by Codebase**

| | Chrome | Firefox | Android |
|---|---|---|---|
| 2009 | 2% | | |
| 2010 | 9.1% | | |
| 2011 | 11% | | |
| 2012 | 10.7% | 14.6% | |
| 2013 | 10.8% | 14.3% | |
| 2014 | 16.5% | 19% | |
| 2015 | 17.6% | 18.9% | 15.4% |
| 2016 | 19.1% | 17.4% | 23.3% |

The fact that these rediscovery rates are trending higher and, in some cases, are two to three times previous estimates, is important. The impact of rediscovery scales with the number of vulnerabilities in a codebase. Looking at Chrome, for example, in 2014 there were 600 recorded high- and critical-severity vulnerabilities. At a rediscovery rate of 6%, closer to Ozment's original estimates, Google should expect that at least 36 vulnerabilities known to them in 2014 will have been discovered by another party, potentially before being disclosed. Using the rate we found above, 16.5%, that number jumps to 99 vulnerabilities.

This means more than 60 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.[41] It bears on the discussion as well that even when there is a patch available, many vulnerabilities don't immediately become useless, as users and organizations often delay applying patching for months or longer.[42] Thus, a patched vulnerability still can still have operational utility.

41 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.

42 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 relatively long lag time between original disclosure and rediscovery, more than two months on average across data from Android and the Chrome browser, suggests that many rediscoveries are truly independent. 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. Because the time lag is so long, it suggests many of these rediscoveries are truly independent and thus a more accurate model of the behavior of malicious third parties.

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, the data in this paper includes software that would have to be part of any such study. Thus, while we can only claim that our estimates are higher than previously reported in scholarship available to the public, these findings should also speak directly to the policy community, regardless of classification.

There are many reasons to believe that the rediscovery rates presented in this paper are an *under*estimate of the true rate of rediscovery. Records from Bugcrowd discussed previously indicate that low- and medium-severity vulnerabilities are rediscovered more frequently than the high- and critical-severity bugs to which this study is constrained. As it is, the 15% to 20% estimate is substantially higher than previously seen. This is a first effort, but one which attempts to bring some empirical work to bear on an issue of interest to both the research and policy communities.

**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