The Secret Life of Software Vulnerabilities

A Large-Scale Empirical Study

Emanuele Iannone, Roberta Guadagni, Filomena Ferrucci, Andrea De Lucia, Fabio Palomba

University of Salerno, Italy

Made in sesa lab
SOFTWARE ENGINEERING SALERNO

eiannone@unisa.it
https://emaiannone.github.io/
@EmanueleIannon3

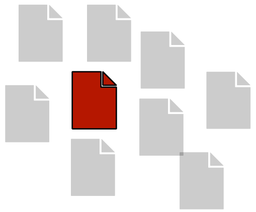
Software Vulnerabilities are…

… okay just kidding, let's start directly with a real-world example!

Let's consider a vulnerability related to the authentication server of *Cloud Foundry*:

CVE-2019-11274

A Cross-Site Scripting vulnerability caused by an unsanitized URL query parameter.

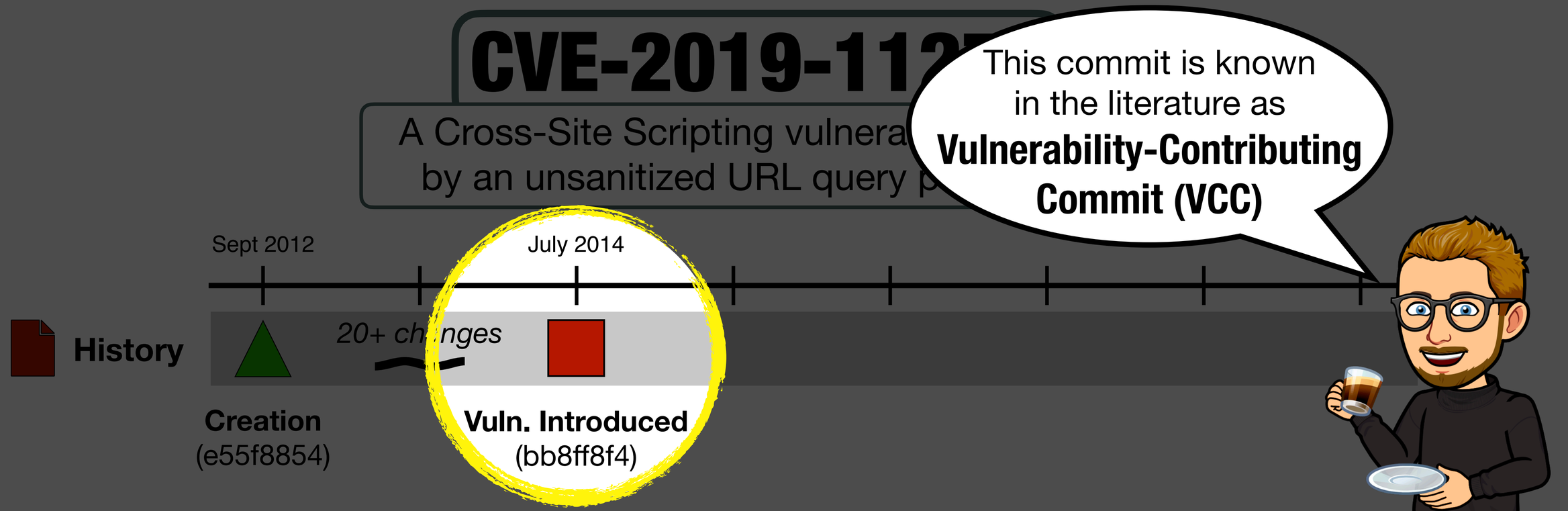


The vulnerability was entirely caused by a flaw in just one file.

Software Vulnerabilities are…

… okay just kidding, let's start directly with a real-world example!

Let's consider a vulnerability related to the authentication server of *Cloud Foundry*:



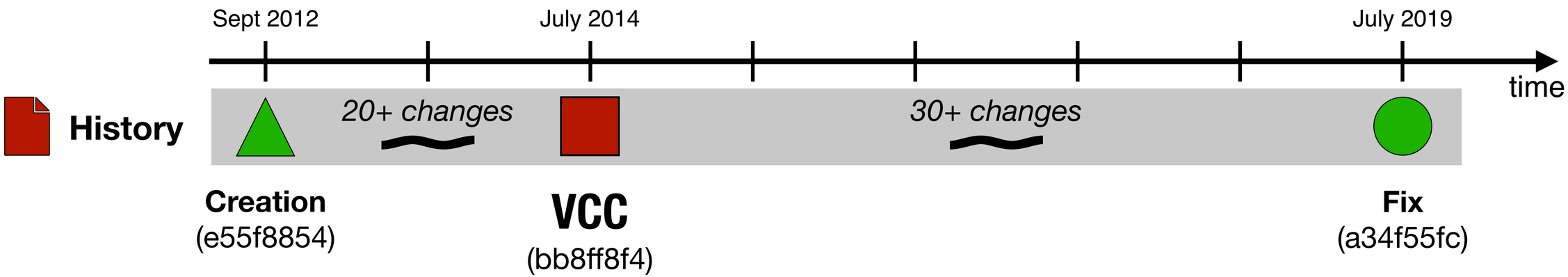
Software Vulnerabilities are…

… okay just kidding, let's start directly with a real-world example!

Let's consider a vulnerability related to the authentication server of *Cloud Foundry*:

CVE-2019-11274

A Cross-Site Scripting vulnerability caused by an unsanitized URL query parameter.



Software Vulnerabilities are…

… okay just kidding, let's start directly with a real-world example!

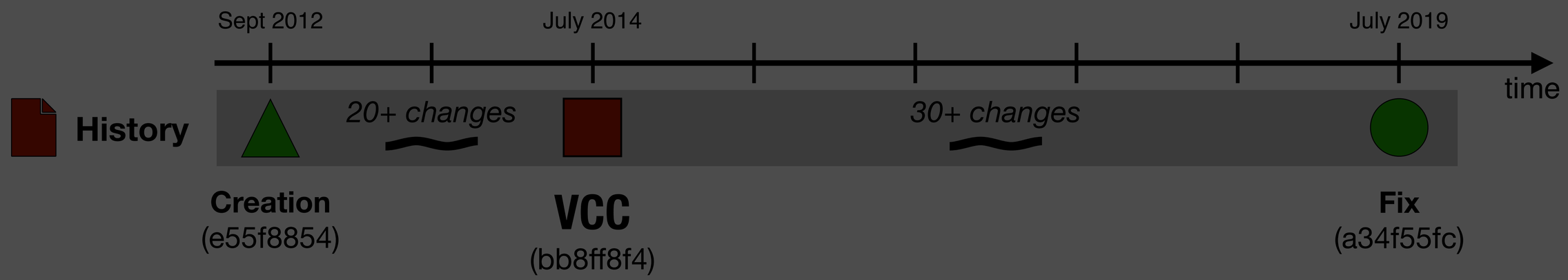
Let's consider a vulnerability related to the authentication server of *Cloud Foundry*:

We have just seen an example of the

Life Cycle of a Vulnerability!

CVE-2016-4037

A cross-site scripting vulnerability caused by an unsanitized URL query parameter.



In literature, we find some empirical studies on the life cycle of vulnerabilities mined from many sources.

Background

Large-Scale Vulnerability Analysis

Stefan Frei, Martin May, Ulrich Fiedler, Bernhard Plattner
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
{stefan.frei, may, fiedler, plattner}@tik.ee.ethz.ch

ABSTRACT

The security level of networks and systems is determined by the software vulnerabilities of its elements. Defending against large scale attacks requires a quantitative understanding of the vulnerability lifecycle. Specifically, one has to understand how exploitation and remediation of vulnerabilities, as well as the distribution of information thereof is handled by industry.

In this paper, we examine how vulnerabilities are handled in large-scale, analyzing more than 80,000 security advisories published since 1995. Based on this information, we quantify the performance of the security industry as a whole. We discover trends and discuss their implications. We quantify the gap between exploit and patch availability and provide an analytical representation of our data which lays the foundation for further analysis and risk management.

Keywords

vulnerability lifecycle, disclosure date, exploit, patch, business risk management, security exposure, intrusion detection, security dynamics

1. INTRODUCTION

It is an accepted fact that most software written gives rise to design and implementation weaknesses. Such flaws may lead to vulnerabilities that potentially open operating systems and applications to attack or misuse. Vulnerabilities are of significant interest when the program containing the flaw operates in a networked environment or has access to the Internet. When vulnerabilities are discovered, disclosed, and exploited, they give rise to individual and large-scale attacks.

The security industry and software vendors try to match the rate of newly discovered vulnerabilities by providing countermeasures such as signatures for viruses, intrusion prevention systems and software patches. To understand the security risks inherent with the use and operation of today's large and complex information and communication systems, analysis of the vulnerabilities' technical details alone is not sufficient. To assess the risk exposure of the network, one has to know and understand the lifecycle of vulnerabilities and the evolution thereof.

The measurement of the cumulated number of disclosed vulnerabilities over time (see Figure 1, [30]) is an interesting indicator of the increasing risk for large scale attacks, but is not sufficient for an analysis thereof. The underlying numbers of such figures contain no information on the time a system is potentially on exposure when the vulnerability is known to the public, or when remediation is available.

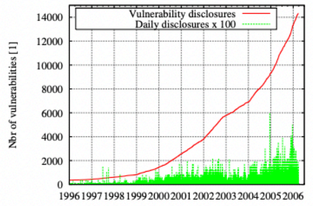


Figure 1: Cumulated number and daily rates of disclosed vulnerabilities between 1996 and 2006

In this paper, we address this problem by examining how vulnerabilities are handled in a large scale. Therefore, we are collecting data of known vulnerabilities and analyze them with regard to information about discovery date, disclosure date, as well as the exploit and patch availability date. Specifically we are looking for answers to the following questions:

1. What is, in a large scale, the relation between the time of discovery and disclosure of vulnerabilities? How relates the time of availability of security patches with the availability of exploits?
2. How responsive is the security industry as a whole with regard to security threats? And how evolves this responsiveness over time.
3. How can we provide the data necessary to perform risk management studies?

For this analysis, we collect information of discovery, exploit availability, and patch availability dates from publicly available sources, such as vulnerability databases and security advisories.

SIGCOMM'06 Workshops September 11-15, 2006, Pisa, Italy.

S. Frei et al., "Large-scale vulnerability analysis", Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense (LSAD '06). Association for Computing Machinery, https://doi.org/10.1145/1162666.1162671

Modelling the Security Ecosystem - The Dynamics of (In)Security

Stefan Frei[1], Dominik Schatzmann[1], Bernhard Plattner[1], Brian Trammell[2]
[1] Communication Systems Group, ETH Zurich, Switzerland
[2] Hitachi Europe, ICTL Secure Systems Team, Zürich, Switzerland

http://www.techzoom.net/security-ecosystem

1. ABSTRACT

The security of information technology and computer networks is effected by a wide variety of actors and processes which together make up a security ecosystem; here we examine this ecosystem, consolidating many aspects of security that have hitherto been discussed only separately. First, we analyze the roles of the major actors within this ecosystem and the processes they participate in, and the the paths vulnerability data take through the ecosystem and the impact of each of these on security risk. Then, based on a quantitative examination of 27,000 vulnerabilities disclosed over the past decade and taken from publicly available data sources, we quantify the systematic gap between exploit and patch availability. We provide the first examination of the impact and the risks associated with this gap on the ecosystem as a whole. Our analysis provides a metric for the success of the "responsible disclosure" process. We measure the prevalence of the commercial markets for vulnerability information and highlight the role of security information providers (SIP), which function as the "free press" of the ecosystem.

2. INTRODUCTION

With the ongoing deployment of information technology in today's economy and society, comprehending the evolution of information security at large has become much more than the mere understanding of the underlying technologies. There is a growing realization that security failures are caused as often by bad incentives as by bad design or neglected implementation: Insecurity often results from what economists call an *externality*, a side-effect of using information technology, like environmental pollution [1]. E.g. vulnerabilities in software impose costs on the whole society of users, while software vendors get all the profits. Whenever a new vulnerability is discovered, various parties with different and often conflicting motives and incentives become engaged in a complex way. These players and their interactions form what we call the *Security Ecosystem*. The security impact resulting from the interplay of the actors of the security ecosystem cannot be understood and managed unless we can better measure these risks.

The goal of this paper is to develop metrics that help to obtain a better understanding of the state and the evolution of today's security environment from a global perspective. Our method to give insight into the dynamics and the prevalence of important processes of the security ecosystem is the analysis of the *Lifecycle of a Vulnerability*, based entirely on publicly available data from various sources. In the following we define the lifecycle of a vulnerability and introduce a model to describe the main players and their interactions in the security ecosystem. The sequence of events in the vulnerability lifecycle measures the main processes governing the security ecosystem. To support the understanding of these complex processes we revisit the key elements of the "disclosure debate", look at "vulnerability markets", and analyze the motivations of vendors and cyber-criminals. Finally we show how the security ecosystem can be described and analyzed quantitatively using statistical analysis of the vulnerability lifecycle.

3. RELATED WORK

After years of providing more and more security features, a realization emerged that a pure technical point of view is not sufficient to understand the ever evolving security landscape [1]. According to [2], the *security ecosystem* describes the activities of creating, preventing, dealing with, and mitigating insecurity in the use of information technology. The economics of information security is *cross-disciplinary* as much as *interdisciplinary* according to Pfleeger [3]. Quantitative measurements of the security ecosystem typically focused on partial analysis of individual events. In "The new school of information security" Shostack and Stewart observe that until today there exist no aggregated long-term indicators or indexes to better understand how the security ecosystem functions [4]. Research on the economic consequences of cyber attacks has been dealing primarily with microanalysis of specific events, technologies or targeted organizations [3, 5]. In 2004, Cavusoglu and Arora examine how a disclosure policy affects the time for a vendor to release a patch [6, 7], and Cavusoglu demonstrates in [8] that synchronization of patch-release and updates cycles minimizes social loss. Kannan and Telang study whether market-based mechanism for vulnerability disclosure lead to a better social outcome [9]. The lure of money is changing the computer

S. Frei et al., "Modeling the Security Ecosystem - The Dynamics of (In)Security", C. (eds) Economics of Information Security and Privacy. Springer, 2010, https://doi.org/10.1007/978-1-4419-6967-5_6

A Large Scale Exploratory Analysis of Software Vulnerability Life Cycles

Muhammad Shahzad, Muhammad Zubair Shafiq, Alex X. Liu
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI, U.S.A.
{*shahzadm, shafiqmu, alexliu*}*@cse.msu.edu*

***Abstract*—Software systems inherently contain vulnerabilities that have been exploited in the past resulting in significant revenue losses. The study of vulnerability life cycles can help in the development, deployment, and maintenance of software systems. It can also help in designing future security policies and conducting audits of past incidents. Furthermore, such an analysis can help customers to assess the security risks associated with software products of different vendors.**

In this paper, we conduct an exploratory measurement study of a large software vulnerability data set containing 46310 vulnerabilities disclosed since 1988 till 2011. We investigate vulnerabilities along following seven dimensions: (1) phases in the life cycle of vulnerabilities, (2) evolution of vulnerabilities over the years, (3) functionality of vulnerabilities, (4) access requirement for exploitation of vulnerabilities, (5) risk level of vulnerabilities, (6) software vendors, and (7) software products. Our exploratory analysis uncovers several statistically significant findings that have important implications for software development and deployment.

***Keywords*-vulnerability; disclosure; patch; exploit; NVD; OSVDB**

I. INTRODUCTION

In computer software, a vulnerability is a loophole in the software code that enables an attacker to circumvent the deployed security measures [1]. Each software vulnerability has a life cycle that consists of distinct phases characterized by the events of its discovery, disclosure, exploitation, and patching. Each phase has a certain level of risk associated with it. The first phase of the life cycle of a vulnerability starts when it is discovered by the vendor, a hacker, or any third-party software analyst. The security risk associated with a vulnerability is particularly high if it is first discovered by hackers. The next phase starts with the public disclosure of the vulnerability, which can again be done by the vendor, a hacker, or any third-party software analyst. After disclosure, the information about a vulnerability is freely available to everyone; therefore, the level of security risk increases further because the hacker community is active in developing and releasing *zero-day exploits* [2]. The aim of the vendor is to release a *patch* for the vulnerability as soon as possible. It is noteworthy that many users of the affected software do not instantly install the patch released to fix the vulnerability. The life cycle of a vulnerability ends when all users of a software install the patch to fix the vulnerability. A vulnerability can be exploited by hackers at any time during its entire life cycle.

The exploratory analysis of vulnerability life cycles can uncover interesting patterns for vendors and software products that are helpful in following ways: First, a thorough analysis is helpful in the deployment of best practices in the software development processes. Second, such analysis is useful to develop the security policies that can handle future attacks and threats more effectively. Third, an exploratory analysis provides insights about the previous security incidents that are helpful in their audit. Finally, it also helps customers to assess the security risks associated with the software products of a particular vendor.

To the best of our knowledge, no previous work has been done to analyze the evolution of life cycle of different types of vulnerabilities for different software products and vendors. The only work in this direction was reported by Frei *et al.* [3], [4]. In [3], Frei *et al.* studied the performance of the software industry as a whole but did not characterize the behavior of individual vendors. In [4], the authors only compared the vulnerability handling process of two vendors and based their analysis on a small data set. Some researchers have focused on the modeling of vulnerability discovery process [2], [5], [6]. The goal of such work is to estimate the number of vulnerabilities in new software products. Another direction of work aims to study the changes in the patching behavior of vendors in response to vulnerability disclosures and the existence of competitors [7], [8]. These studies analyze only small vulnerability data sets and do not cover the behavior of individual vendors.

In this paper we make following three contributions. (1) We have aggregated a large software vulnerability data set from three vulnerability repositories: (a) National Vulnerability Database (NVD) [9], (b) Open Source Vulnerability Database (OSVDB) [10], and (c) the vulnerability data collected by Frei *et al.* (FVDB) [3]. Our aggregated software vulnerability data set contains 46310 vulnerabilities since 1988 to 2011. (2) We have comprehensively analyzed software vulnerabilities along the seven dimensions mentioned in the abstract. Our observations are supported by statistical tests for significance. (3) To systematically analyze patterns in our vulnerability data set, we have utilized association rule mining to extract rules that represent exploitation behavior of hackers and the patching behavior of vendors.

M. Shahzad et al., "A large scale exploratory analysis of software vulnerability life cycles," 2012 34th International Conference on Software Engineering (ICSE), Zurich, Switzerland, 2012, pp. 771-781, doi: 10.1109/ICSE.2012.6227141.

Background

In literature, we find some empirical studies on the life cycle of vulnerabilities mined from many sources.

Discovery

Disclosure

Exploitation

Patching

Large-Scale Vulnerability Analysis

Stefan Frei, Martin May, Ulrich Fie
Computer Engineering and Ne
ETH Zurich, Switz
{stefan.frei, may, fiedler, platt

ABSTRACT

The security level of networks and systems is determined by the software vulnerabilities of its elements. Defending against large scale attacks requires a quantitative understanding of the vulnerability lifecycle. Specifically, one has to understand how exploitation and remediation of vulnerabilities, as well as the distribution of information thereof is handled by industry.

In this paper, we examine how vulnerabilities are handled in large-scale, analyzing more than 80,000 security advisories published since 1995. Based on this information, we quantify the performance of the security industry as a whole. We discover trends and discuss their implications. We quantify the gap between exploit and patch availability and provide an analytical representation of our data which lays the foundation for further analysis and risk management.

Keywords

vulnerability lifecycle, disclosure date, exploit, patch, business risk management, security exposure, intrusion detection, security dynamics

1. INTRODUCTION

It is an accepted fact that most software written gives rise to design and implementation weaknesses. Such flaws may lead to vulnerabilities that potentially open operating systems and applications to attack or misuse. Vulnerabilities are of significant interest when the program containing the flaw operates in a networked environment or has access to the Internet. When vulnerabilities are discovered, disclosed, and exploited, they give rise to individual and large-scale attacks.

The security industry and software vendors try to match the rate of newly discovered vulnerabilities by providing countermeasures such as signatures for viruses, intrusion prevention systems and software patches. To understand the security risks inherent with the use and operation of today's large and complex information and communication systems, analysis of the vulnerabilities' technical details alone is not sufficient. To assess the risk exposure of

SIGCOMM'06 Workshops September 11-15, 2006, Pisa, Italy.

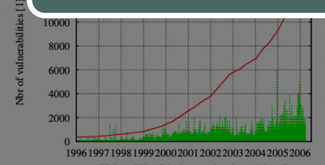


Figure 1: Cumulated number and daily rates of disclosed vulnerabilities between 1996 and 2006

In this paper, we address this problem by examining how vulnerabilities are handled in a large scale. Therefore, we are collecting data of known vulnerabilities and analyze them with regard to information about discovery date, disclosure date, as well as the exploit and patch availability date. Specifically we are looking for answers to the following questions:

1. What is, in a large scale, the relation between the time of discovery and disclosure of vulnerabilities? How relates the time of availability of security patches with the availability of exploits?
2. How responsive is the security industry as a whole with regard to security threats? And how evolves this responsiveness over time.
3. How can we provide the data necessary to perform risk management studies?

For this analysis, we collect information of discovery, exploit availability, and patch availability dates from publicly available sources, such as vulnerability databases and security advisories.

S. Frei et al., "Large-scale vulnerability analysis", Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense (LSAD '06). Association for Computing Machinery, https://doi.org/10.1145/1162666.1162671

Modelling the Security Ecosystem
The Dynamics of (In)Security

atzmann[1], Bernhard Plattner[1], Brian Trammell[2]
ystems Group, ETH Zurich, Switzerland
Secure Systems Team, Zürich, Switzerland

echzoom.net/security-ecosystem

computer
s and pro-
ecosystem;
many as-
ssed only
major ac-
they par-
data take
h of these
ive exam-
r the past
data sources,
we quantify the systematic gap between exploit and patch availability. We provide the first examination of the impact and the risks associated with this gap on the ecosystem as a whole. Our analysis provides a metric for the success of the "responsible disclosure" process. We measure the prevalence of the commercial markets for vulnerability information and highlight the role of security information providers (SIP), which function as the "free press" of the ecosystem.

2. INTRODUCTION

With the ongoing deployment of information technology in today's economy and society, comprehending the evolution of information security at large has become much more than the mere understanding of the underlying technologies. There is a growing realization that security failures are caused as often by bad incentives as by bad design or neglected implementation: Insecurity often results from what economists call an *externality*, a side-effect of using information technology, like environmental pollution [1]. E.g. vulnerabilities in software impose costs on the whole society of users, while software vendors get all the profits. Whenever a new vulnerability is discovered, various parties with different and often conflicting motives and incentives become engaged in a complex way. These players and their interactions form what we call the *Security Ecosystem*. The security impact resulting from the interplay of the actors of the security ecosystem cannot be understood and managed unless we can better measure these risks.

The goal of this paper is to develop metrics that help to obtain a better understanding of the state and the evolution of today's security environment from a global perspective. Our method to give insight into the dynamics and the prevalence of important processes of the security ecosystem is the analysis of the *Lifecycle of a Vulnerability*, based entirely on publicly available data from various sources. In the following we define the lifecycle of a vulnerability and introduce a model to describe the main players and their interactions in the security ecosystem. The sequence of events in the vulnerability lifecycle measures the main processes governing the security ecosystem. To support the understanding of these complex processes we revisit the key elements of the "disclosure debate", look at "vulnerability markets", and analyze the motivations of vendors and cyber-criminals. Finally we show how the security ecosystem can be described and analyzed quantitatively using statistical analysis of the vulnerability lifecycle.

3. RELATED WORK

After years of providing more and more security features, a realization emerged that a pure technical point of view is not sufficient to understand the ever evolving security landscape [1]. According to [2], the *security ecosystem* describes the activities of creating, preventing, dealing with, and mitigating insecurity in the use of information technology. The economics of information security is *cross-disciplinary* as much as *interdisciplinary* according to Pfleeger [3]. Quantitative measurements of the security ecosystem typically focused on partial analysis of individual events. In "The new school of information security" Shostack and Stewart observe that until today there exist no aggregated long-term indicators or indexes to better understand how the security ecosystem functions [4]. Research on the economic consequences of cyber attacks has been dealing primarily with microanalysis of specific events, technologies or targeted organizations [3, 5]. In 2004, Cavusoglu and Arora examine how a disclosure policy affects the time for a vendor to release a patch [6, 7], and Cavusoglu demonstrates in [8] that synchronization of patch-release and updates cycles minimizes social loss. Kannan and Telang study whether market-based mechanism for vulnerability disclosure lead to a better social outcome [9]. The lure of money is changing the computer

S. Frei et al., "Modeling the Security Ecosystem - The Dynamics of (In)Security", C. (eds) Economics of Information Security and Privacy. Springer, 2010, https://doi.org/10.1007/978-1-4419-6967-5_6

A Large Scale Exploratory Analysis of Software Vulnerability Life Cycles

Muhammad Shahzad, Muhammad Zubair Shafiq, Alex X. Liu
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI, U.S.A.
{*shahzadm, shafiqmu, alexliu*}*@cse.msu.edu*

***Abstract*—Software systems inherently contain vulnerabilities that have been exploited in the past resulting in significant revenue losses. The study of vulnerability life cycles can help in the development, deployment, and maintenance of software systems. It can also help in designing future security policies and conducting audits of past incidents. Furthermore, such an analysis can help customers to assess the security risks associated with software products of different vendors.**

In this paper, we conduct an exploratory measurement study of a large software vulnerability data set containing 46310 vulnerabilities disclosed since 1988 till 2011. We investigate vulnerabilities along following seven dimensions: (1) phases in the life cycle of vulnerabilities, (2) evolution of vulnerabilities over the years, (3) functionality of vulnerabilities, (4) access requirement for exploitation of vulnerabilities, (5) risk level of vulnerabilities, (6) software vendors, and (7) software products. Our exploratory analysis uncovers several statistically significant findings that have important implications for software development and deployment.

***Keywords*-vulnerability; disclosure; patch; exploit; NVD; OSVDB**

I. Introduction

In computer software, a vulnerability is a loophole in the software code that enables an attacker to circumvent the deployed security measures [1]. Each software vulnerability has a life cycle that consists of distinct phases characterized by the events of its discovery, disclosure, exploitation, and patching. Each phase has a certain level of risk associated with it. The first phase of the life cycle of a vulnerability starts when it is discovered by the vendor, a hacker, or any third-party software analyst. The security risk associated with a vulnerability is particularly high if it is first discovered by hackers. The next phase starts with the public disclosure of the vulnerability, which can again be done by the vendor, a hacker, or any third-party software analyst. After disclosure, the information about a vulnerability is freely available to everyone; therefore, the level of security risk increases further because the hacker community is active in developing and releasing *zero-day exploits* [2]. The aim of the vendor is to release a *patch* for the vulnerability as soon as possible. It is noteworthy that many users of the affected software do not instantly install the patch released to fix the vulnerability. The life cycle of a vulnerability ends when all users of a software install the patch to fix the vulnerability. A vulnerability can be exploited by hackers at any time during its entire life cycle.

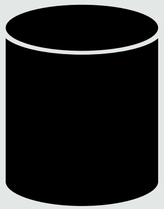
The exploratory analysis of vulnerability life cycles can uncover interesting patterns for vendors and software products that are helpful in following ways: First, a thorough analysis is helpful in the deployment of best practices in the software development processes. Second, such analysis is useful to develop the security policies that can handle future attacks and threats more effectively. Third, an exploratory analysis provides insights about the previous security incidents that are helpful in their audit. Finally, it also helps customers to assess the security risks associated with the software products of a particular vendor.

To the best of our knowledge, no previous work has been done to analyze the evolution of life cycle of different types of vulnerabilities for different software products and vendors. The only work in this direction was reported by Frei *et al.* [3], [4]. In [3], Frei *et al.* studied the performance of the software industry as a whole but did not characterize the behavior of individual vendors. In [4], the authors only compared the vulnerability handling process of two vendors and based their analysis on a small data set. Some researchers have focused on the modeling of vulnerability discovery process [2], [5], [6]. The goal of such work is to estimate the number of vulnerabilities in new software products. Another direction of work aims to study the changes in the patching behavior of vendors in response to vulnerability disclosures and the existence of competitors [7], [8]. These studies analyze only small vulnerability data sets and do not cover the behavior of individual vendors.

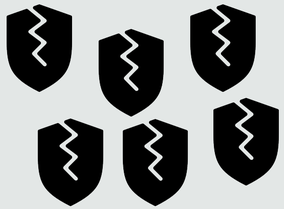
In this paper we make following three contributions. (1) We have aggregated a large software vulnerability data set from three vulnerability repositories: (a) National Vulnerability Database (NVD) [9], (b) Open Source Vulnerability Database (OSVDB) [10], and (c) the vulnerability data collected by Frei *et al.* (FVDB) [3]. Our aggregated software vulnerability data set contains 46310 vulnerabilities since 1988 to 2011. (2) We have comprehensively analyzed software vulnerabilities along the seven dimensions mentioned in the abstract. Our observations are supported by statistical tests for significance. (3) To systematically analyze patterns in our vulnerability data set, we have utilized association rule mining to extract rules that represent exploitation behavior of hackers and the patching behavior of vendors.

M. Shahzad et al., "A large scale exploratory analysis of software vulnerability life cycles," 2012 34th International Conference on Software Engineering (ICSE), Zurich, Switzerland, 2012, pp. 771-781, doi: 10.1109/ICSE.2012.6227141.

Large-scale studies that focus on the insertion-fixing cycle do not exist. We want to shed light on the life of a vulnerability by looking directly at the affected software product.



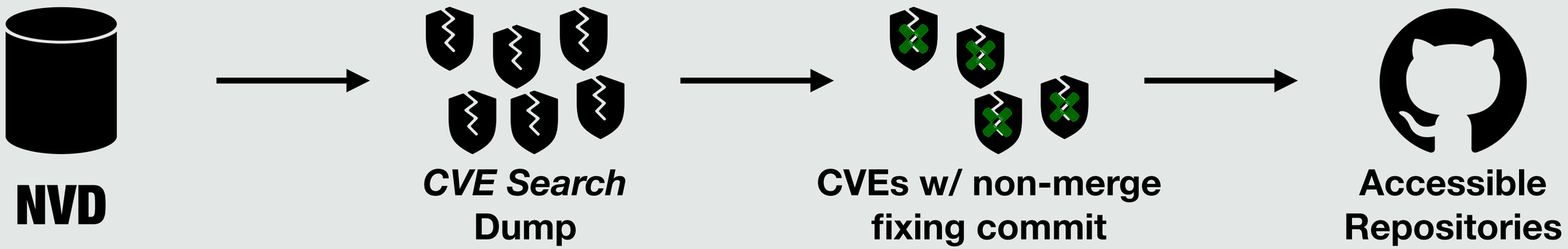
NVD



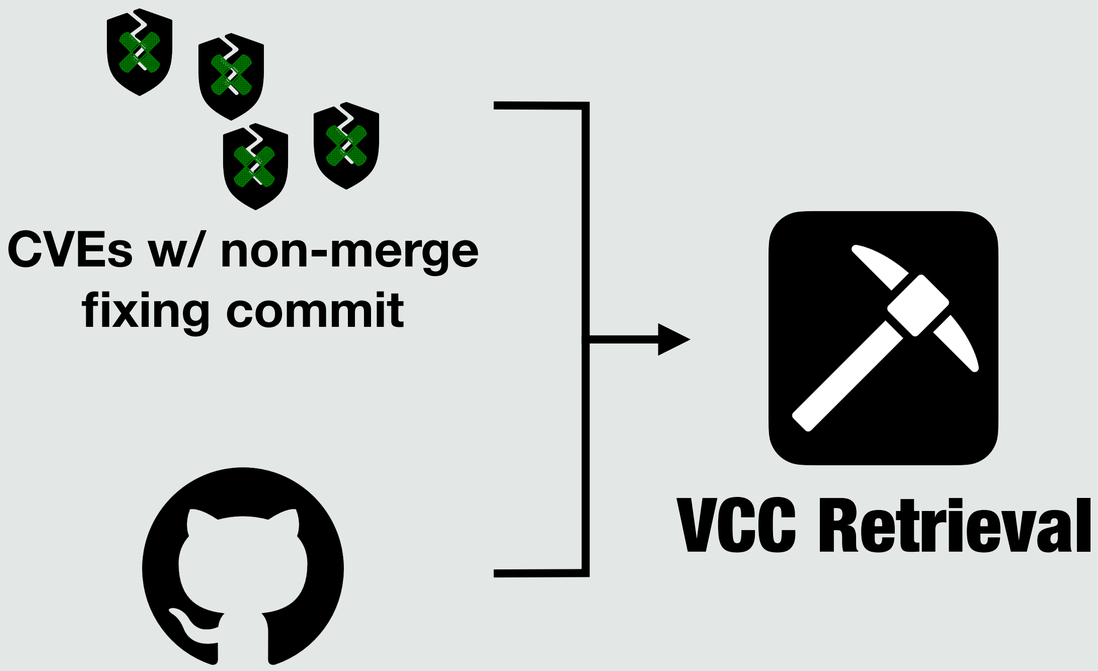
CVE Search
Dump

We downloaded the full content in Nov 2020: more than 142k CVEs.

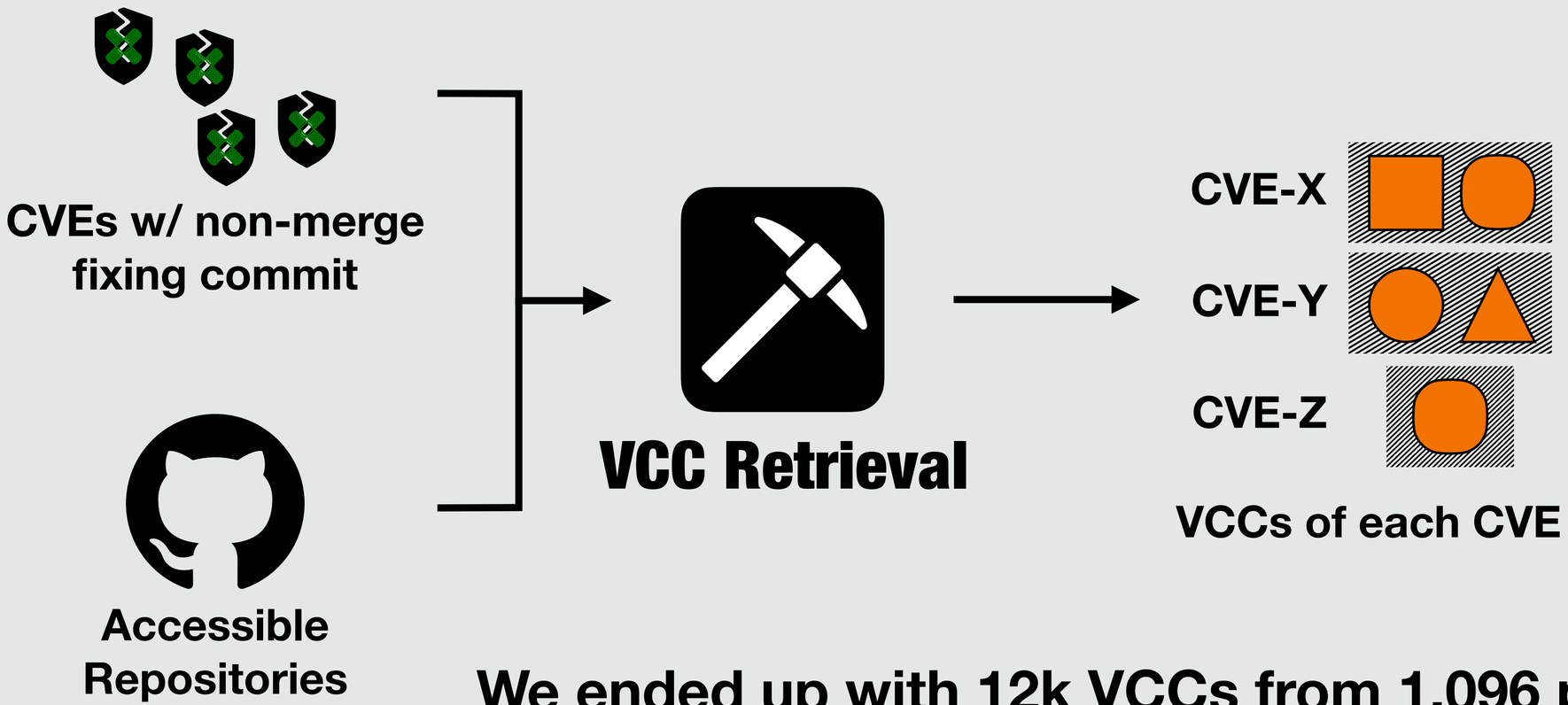
We selected the CVEs having reference to fixing commits: 3,663 CVEs.



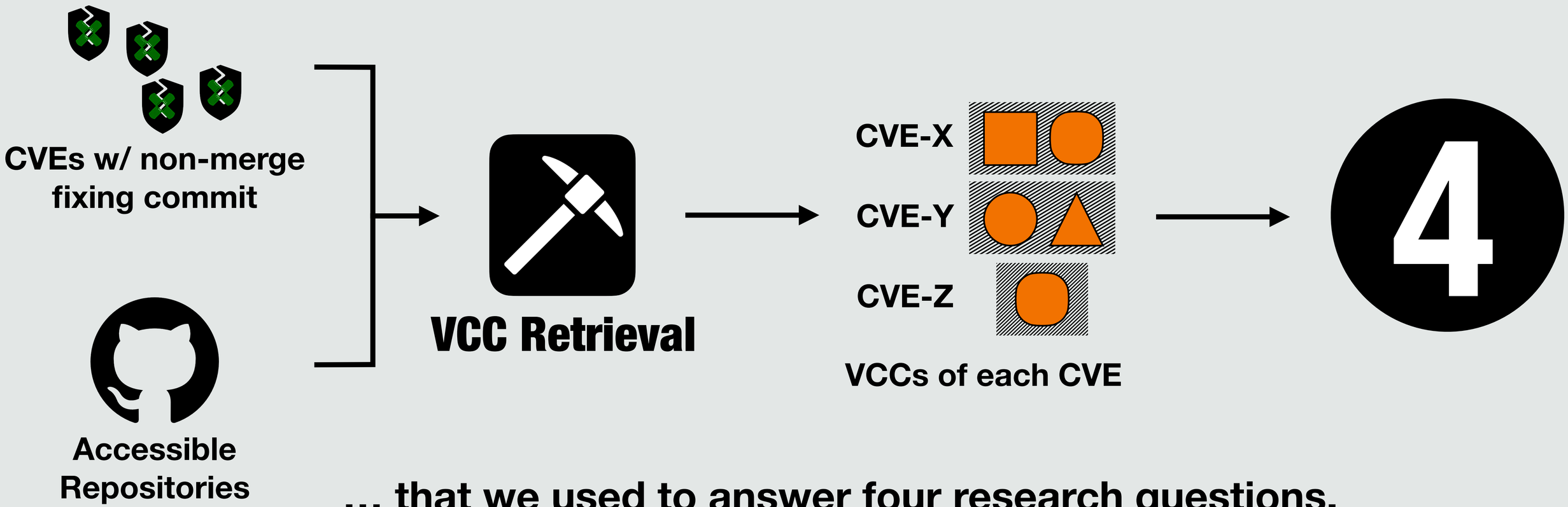
We ensured the linked repositories were still accessible.



We run an algorithm based on a variant SZZ for vulnerabilities to retrieve its VCCs.



We ended up with 12k VCCs from 1,096 projects… that we used to answer four research questions.



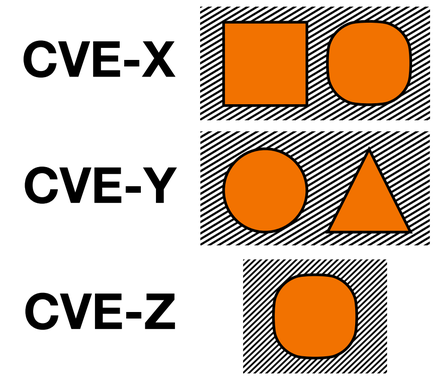
… that we used to answer four research questions.

Results

*How are **contributions** to vulnerabilities made into the source code?*

RQ1

*How are **contributions** to vulnerabilities made into the source code?*



VCCs of each CVE

RQ1

For all CVEs

1 → 2 → ••• → *n*

Nr. VCCs and days for the full introduction

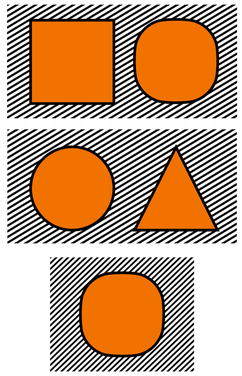
RQ1

*How are **contributions** to vulnerabilities made into the source code?*

CVE-X

CVE-Y

CVE-Z



VCCs of each CVE

RQ1

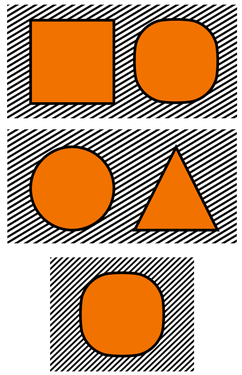
For all CVEs

Most vulnerabilities (~60%) required multiple VCCs and days to be introduced.
On average, at least 4 VCCs over 4 years!

RQ1

*How are **contributions** to vulnerabilities made into the source code?*

CVE-X
CVE-Y
CVE-Z

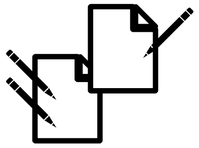


VCCs of each CVE

RQ1

For all CVEs

Most vulnerabilities (~60%) required multiple VCCs and days to be introduced.
On average, at least 4 VCCs over 4 years!

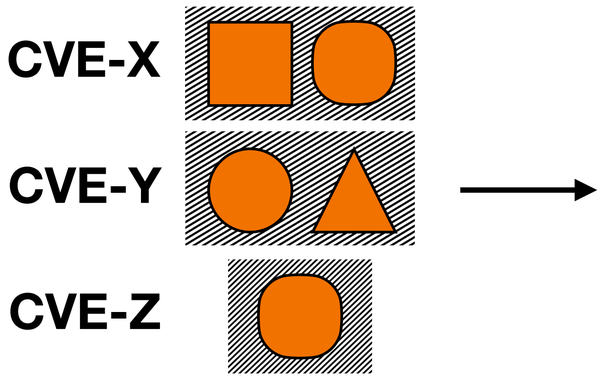


Amount of changes made by VCCs

RQ1

*How are **contributions** to vulnerabilities made into the source code?*

RQ1



VCCs of each CVE

For all CVEs

Most vulnerabilities (~60%) required multiple VCCs and days to be introduced.
On average, at least 4 VCCs over 4 years!

VCCs touch a few files: just one on average. Moreover, they tend to add new lines more than delete them (110 vs. 26).

RQ1

*How are **contributions** to vulnerabilities made into the source code?*

Results

RQ1

For all CVEs

Most vulnerabilities (~60%) required multiple VCCs and days to be introduced

On average, at least 4 VCCs over 4 years!

Vulnerability detection tools could intercept emerging vulnerabilities as soon as the first signals appear, rather than waiting their “final form”.

VCCs touch a few files: just one on average. Moreover, they tend to add new lines more than delete them (110 vs. 26).

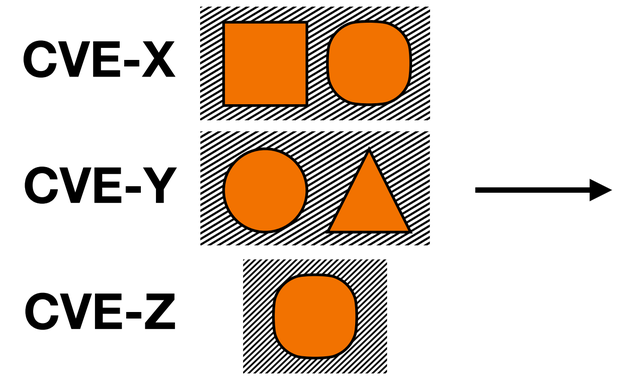
RQ1

*What is the **context** in which contributions are made into the source code?*

RQ2

*What is the **context** in which contributions are made into the source code?*

RQ2



VCCs of each CVE

For all VCCs

We assigned a set of *categories* according to specific characteristics.

Commit Goal (from the Message)

- New feature
- Bug Fixing
- Enhancement
- Refactoring

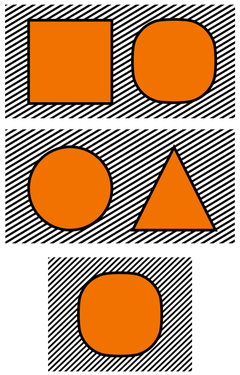
RQ2

*What is the **context** in which contributions are made into the source code?*

CVE-X

CVE-Y

CVE-Z



VCCs of each CVE

RQ2

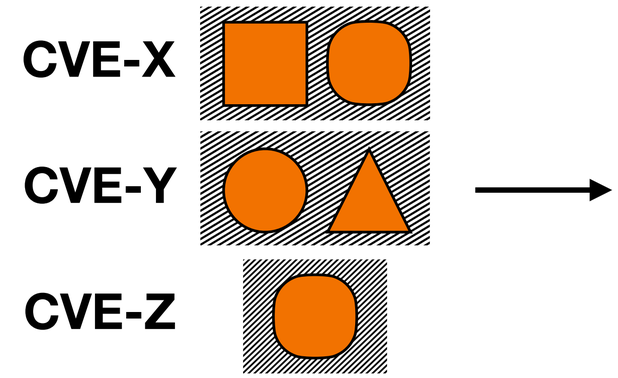
For all VCCs

We assigned a set of *categories* according to specific characteristics.

Most VCCs (~70%) had the explicit goal of making some maintenance activity, such as bug fixing, refactoring, or general enhancements of existing code.

RQ2

*What is the **context** in which contributions are made into the source code?*



VCCs of each CVE

RQ2

For all VCCs

We assigned a set of *categories* according to specific characteristics.

Most VCCs (~70%) had the explicit goal of making some maintenance activity, such as bug fixing, refactoring, or general enhancements of existing code.

Proximity to Release

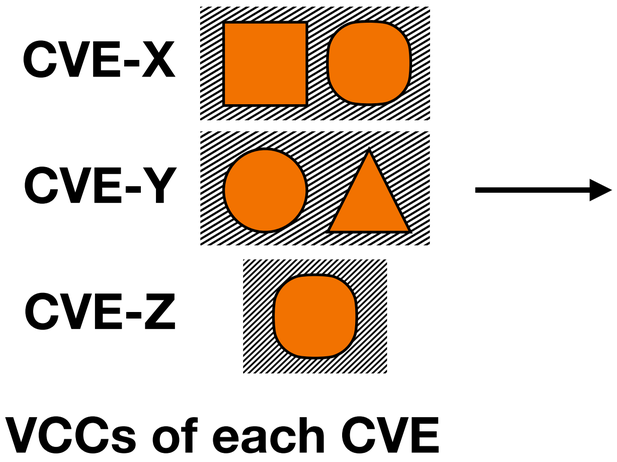
Distance from Project Start

Project Status

RQ2

*What is the **context** in which contributions are made into the source code?*

RQ2



For all VCCs

We assigned a set of *categories* according to specific characteristics.

Most VCCs (~70%) had the explicit goal of making some maintenance activity, such as bug fixing, refactoring, or general enhancements of existing code.

Almost all vulnerabilities (~85%) appeared after the project's first year. More than 60% of VCCs are 30+ days distant from a release.

RQ2

*What is the **context** in which contributions are made into the source code?*

Results

RQ2

For all VCCs

We assigned a set of *categories* according to specific characteristics.

CVE-X

CVE-Y

VCCs of each CVE

Most VCCs (~70%) had the explicit goal of making some [illegible] the vulnerable code.

Contextual information about the files' history and the project status could benefit vulnerability detection tools.

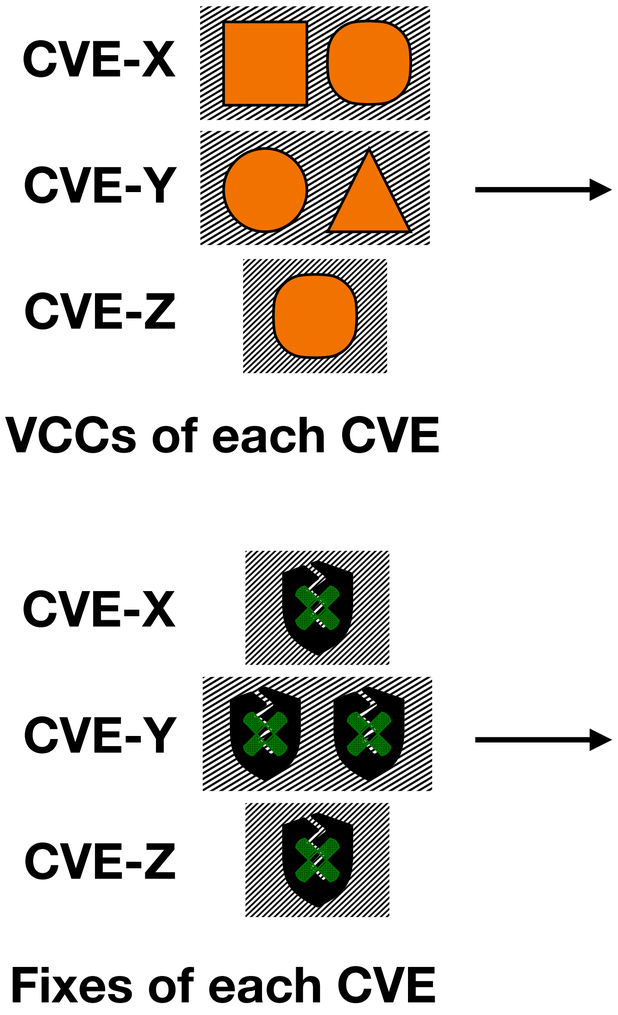
Almost all vulnerabilities (~85%) appeared after the project's first year. More than 60% of VCCs are 30+ days distant from a release.

RQ2

*What is the **survivability** of vulnerabilities?*

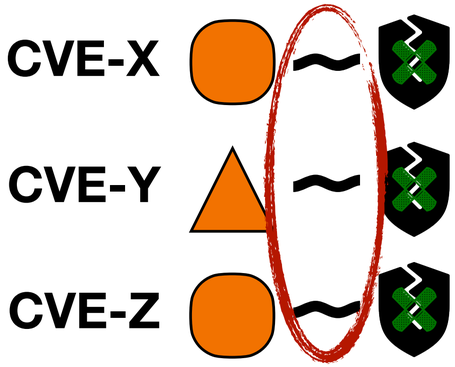
RQ3

*What is the **survivability** of vulnerabilities?*



RQ3

For all CVEs



We analyzed the period between the last VCC and the last fixing commit.

We relied on the Kaplan-Meier estimator to measure their survival probability.

We measured the time in terms of:

Days

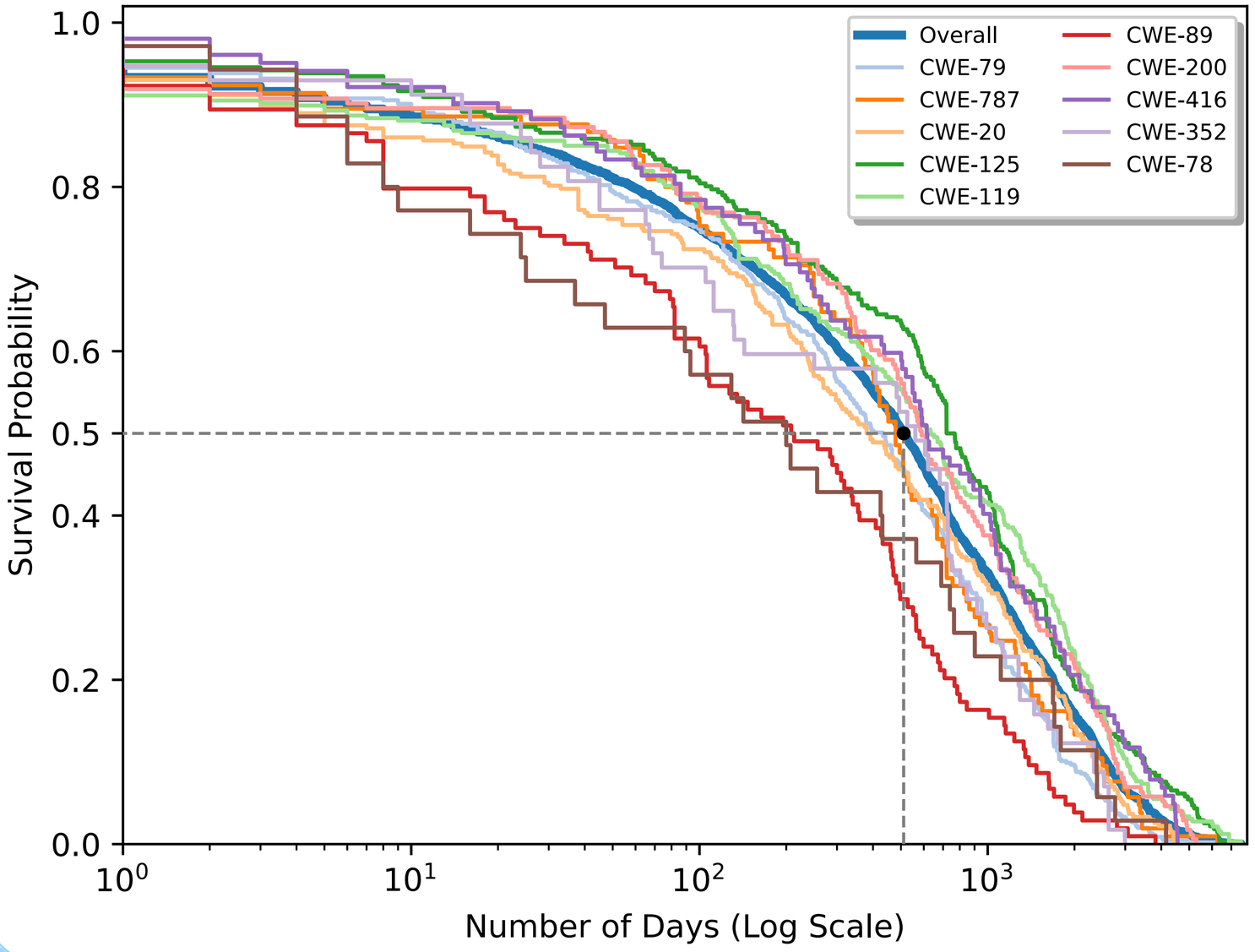
Commits (touching the vulnerable files)

RQ3

Kaplan, E. L., and Paul Meier. “Nonparametric Estimation from Incomplete Observations.” Journal of the American Statistical Association, vol. 53, no. 282, 1958, pp. 457–81. JSTOR, https://doi.org/10.2307/2281868.

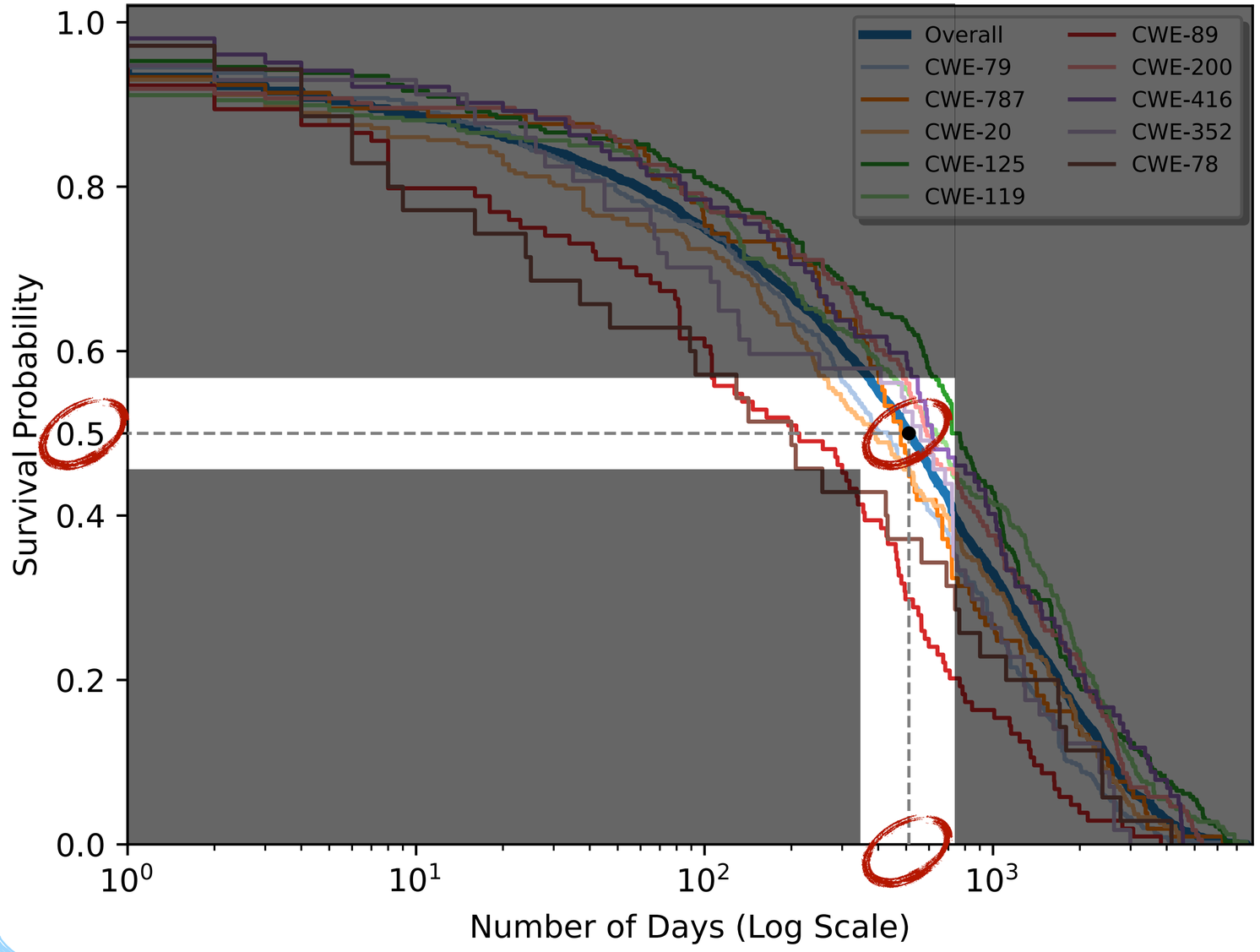
*What is the **survivability** of vulnerabilities?*

RQ3



*What is the **survivability** of vulnerabilities?*

RQ3

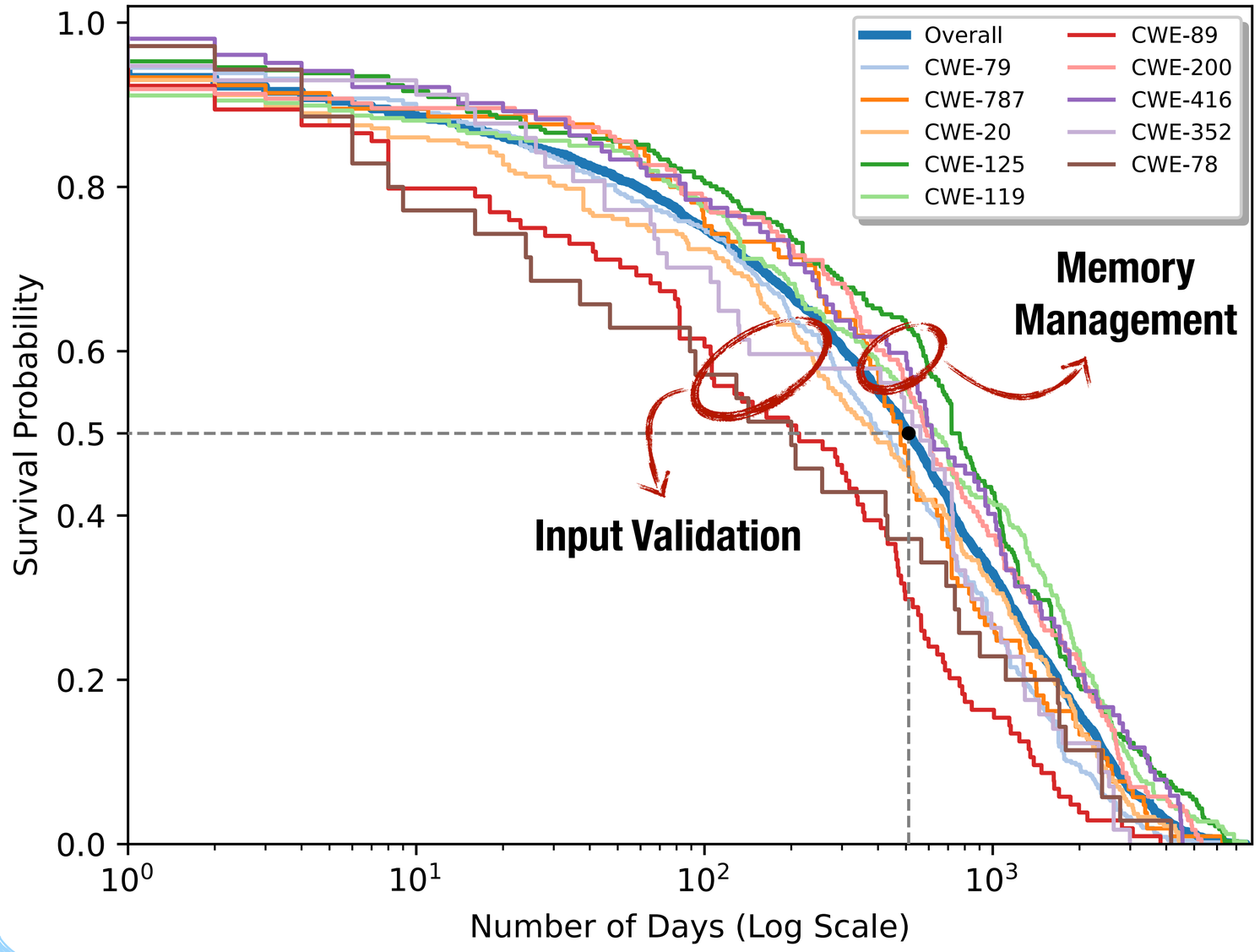


1 out of 2 vulnerabilities survive for at least 511 days (one year and a half)!

RQ3

*What is the **survivability** of vulnerabilities?*

RQ3



1 out of 2 vulnerabilities survive for at least 511 days (one year and a half)!

Input-validation vulnerabilities (CWE-79, CWE-20, etc.) tend to be fixed earlier than others.

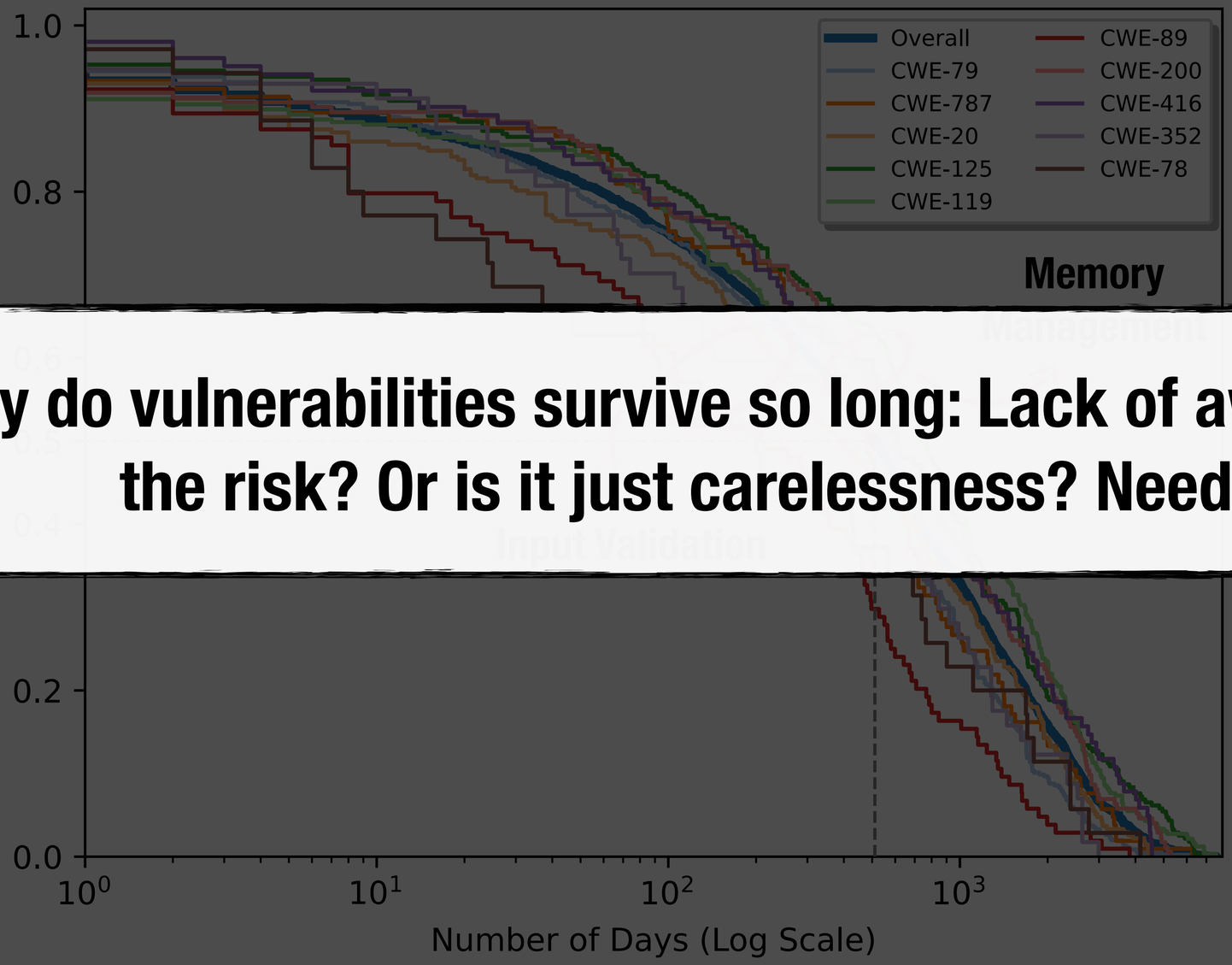
Memory-related vulnerabilities (CWE-787, CWE-125, etc.) tend to be fixed later than others.

RQ3

*What is the **survivability** of vulnerabilities?*

Results

RQ3



1 out of 2 vulnerabilities survive for at least 511 days (one year and a half)!

Why do vulnerabilities survive so long: Lack of awareness? Underestimation of the risk? Or is it just carelessness? Need further investigations.

Memory-related vulnerabilities (CWE-787, CWE-125, etc.) tend to be fixed later than others.

RQ3

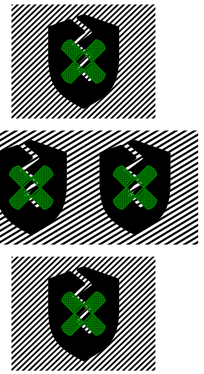
*How are vulnerabilities **removed** from the source code?*

RQ4

*How are vulnerabilities **removed** from the source code?*

RQ4

CVE-X
CVE-Y
CVE-Z



Fixes of each CVE

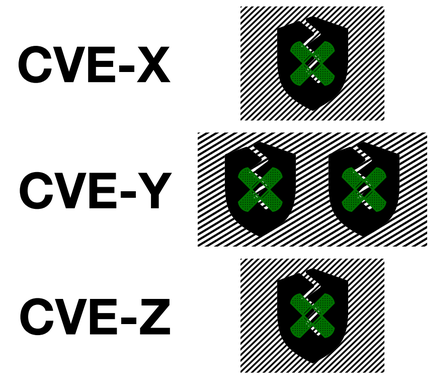
For 351 CVE-Fix pairs

Statistically significant sample!

Determine the kind of fixing action according to two independent inspectors.

Derive a taxonomy of recurring fixing actions.

*How are vulnerabilities **removed** from the source code?*



Fixes of each CVE

RQ4

The inspectors found 23 recurring activities belonging to four different categories.

- **Memory Management**
- **Implementation**
- **Users & Networking**
- **Configuration**

In any case, the fixing commits are relatively simple: On a median, only one file requires fixes, adding 10 new lines and removing four.

*How are vulnerabilities **removed** from the source code?*

RQ4

The inspectors found 23 recurring activities belonging to four different categories.

Fixing vulnerabilities seems not to be the problem. Developers should just be encouraged to apply the recommendations proposed by APR tools.

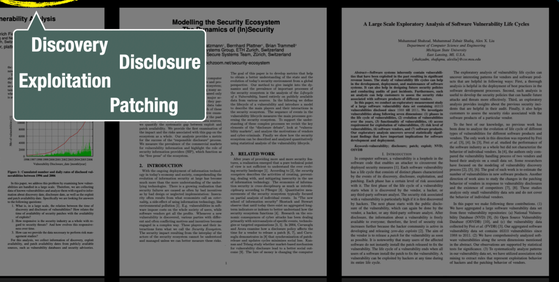
In any case, the fixing commits are relatively simple: On a median, only one file requires fixes, adding 10 new lines and removing four.

The Secret Life of Software Vulnerabilities

A Large-Scale Empirical Study

Background

In literature, we find some empirical studies on the life cycle of vulnerabilities mined from many sources.

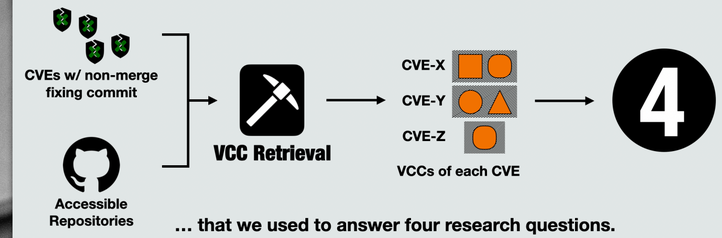


S. Frei et al., "Large-scale vulnerability analysis", Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense (LSAD '06). Association for Computing Machinery, https://doi.org/10.1145/1162666.1162671

S. Frei et al., "Modeling the Security Ecosystem - The Dynamics of (In)Security", C. (eds) Economics of Information Security and Privacy. Springer, 2010, https://doi.org/10.1007/978-1-4419-6967-5_6

M. Shahzad et al., "A large scale exploratory analysis of software vulnerability life cycles," 2012 34th International Conference on Software Engineering (ICSE), Zurich, Switzerland, 2012, pp. 771-781, doi: 10.1109/ICSE.2012.6227141.

Method



… that we used to answer four research questions.

*How are **contributions** to vulnerabilities made into the source code?*

Results

RQ1

For all CVEs

Most vulnerabilities (~60%) required multiple VCCs and days to be introduced

VCCs touch a few files: just one on average. Moreover, they tend to add new lines more than delete them (110 vs. 26).

Vulnerability detection tools could intercept emerging vulnerabilities as soon as the first signals appear, rather than waiting their "final form".

*What is the **context** in which contributions are made into the source code?*

Results

RQ2

For all VCCs

We assigned a set of *categories* according to specific characteristics.

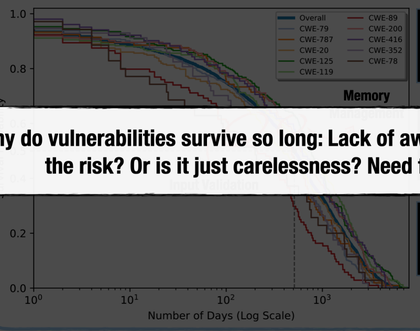
Almost all vulnerabilities (~85%) appeared after the project's first year. More than 60% of VCCs are 30+ days distant from a release.

Contextual information about the files' history and the project status could benefit vulnerability detection tools.

*What is the **survivability** of vulnerabilities?*

Results

RQ3



1 out of 2 vulnerabilities survive for at least 511 days (one year and a half)!

Memory-related vulnerabilities (CWE-787, CWE-125, etc.) tend to be fixed later than others.

Why do vulnerabilities survive so long: Lack of awareness? Underestimation of the risk? Or is it just carelessness? Need further investigations.

*How are vulnerabilities **removed** from the source code?*

Results

RQ4

The inspectors found 23 recurring activities belonging to four different categories.

In any case, the fixing commits are relatively simple: On a median, only one file requires fixes, adding 10 new lines and removing four.

Fixing vulnerabilities does not seem to be the problem. Developers should just be encouraged to apply the recommendations by APR tools.

Thank You!

Questions?