DTU

Technical University of Denmark

# Cyber Security Certifications

What to expect from certifications in the EU Cybersecurity Act and how to assess the security of ICT systems in the meantime?

**Part 2**

Christian Damsgaard Jensen
DTU Compute
Master of Cyber Security Whitepaper 2020-01 (Part 2)

Technical
University
of Denmark

# Table of Contents

# Introduction

This is the second part of the whitepaper that examines the certification of cybersecurity. In the first part of this whitepaper, we examined different approaches to security certifications and discussed possible implementation of security certification in the context of the new EU Cybersecurity Act. In this Part 2, we examine what we can do in the absence of security certified products, i.e. how to assess the security of externally developed software systems. In particular, we explore how to assess the security of a software product or service based on externally available (open source intelligence) data. Such assessment is possible without the active participation of the prospective software or service provider, but it is obviously better if they collaborate in the process and provide the data that are necessary for the evaluation.

Assessing the security of software components is a highly specialized task, which requires considerable resources. These are the main reasons why few organizations perform a complete security assessment of all the components that they come to rely on in their system architecture. The high cost of security assessment is also one of the main reasons why few systems get security certified, as mentioned in Part 1. We therefore need simple methods to provide rough estimates of the security of system components.

In Part 1, we learned that security certification typically focus on one of the following: Products, Processes, or People, so it is natural to consider all three aspects when assessing the security of software products and services. However, we cannot assume full knowledge about the software or service providers' processes or employees (people) so we need to base the security assessment of these on the software developer's track record. This assessment may be based on information, such as news reports, vulnerability disclosures, reputation systems, recommendations from colleagues, professional discussion fora and social media.

# Software Security

Modern software systems and services are generally composed of separate components from multiple suppliers using a common framework or software platform, e.g. using Microsoft's. Net framework. This simplifies the individual components and improves their maintainability, but multiple components also contributes to the complexity of the overall system. In particular, it increases the number of internal and external interfaces between components developed by separate software development teams. An error in any component or disagreement between development teams on either the interpretation or implementation of the specification may lead to a potential security vulnerability. The security assessment of a software component must therefore consider not only the component, but also all the dependencies that each component has.

We therefore divide our security assessment into a Software Security Assessment, which examines both the observable software artefacts and the security track record of the component, and a Company Security Assessment, which provides an indirect assessment of the organizations commitment and ability to develop secure software and systems. The company security assessment considers any available information about the security team as well as the security track record of the organization; in both cases based on open source intelligence data about the provider of the software product or service.

## Software Security Assessment

The availability of source code plays a defining role in all software assessments, so we distinguish between three different cases: all source code available, some source code available, and no source code available.

## All Source Code Available

The source code for all (or most) components and services are typically available for assessment, which is typically the case for internally developed systems and open source software.

Software related metrics for quality software have been studied extensively in the literature, especially in the context of maintainability and dependability. The maintainability attributes are particularly relevant, because they aim to measure the size and complexity of the developed software and complexity is often consider the mother of all security vulnerabilities.

The simplest way to measure the complexity of a software component is to measure its size. This is most commonly done by counting the lines of code in the software, because smaller programs are often easier to comprehend, but several extensions to this metric are possible. In effective lines of code, lines consisting only of comments, blank lines, and lines with standalone brackets are ignored. The logical lines of codes counts the number of statements in the programming language, e.g. lines ending with a semicolon in "Java" or "C". The number of dependencies is another simple metric, which indicates the number of interfaces where divergent interpretations of the specification may surface, i.e. components with fewer dependencies would often be considered more secure. Finally, the comment to code ratio is another simple metric that measures the amount of actual code to the amount of information included to assist software developers. A higher comment to code ratio provides more information for an internal critical code review as well as increase the maintainability of the software.

There are a number of more advanced metrics for code complexity, such as Cyclomatic Complexity [1], the Halsted Complexity [2], and the ABC metric [3], but these generally require tool support, so we do not consider them here.

## Some Source Code Available

Source code is normally not available for commercial software and services, but many systems are built using open source software libraries and frameworks distributed under less restrictive licenses. This means that source code will be available for parts of the system's components and dependencies. These components may then be examined using some of the simple metrics described above, but they may also be treated as black boxes and simply counted as dependencies.

Each component that appears as a dependency embody the assumptions and domain knowledge of the team that developed the component. This means that unless the full semantics of all dependencies are well understood, there is a significant risk of misunderstandings, which may lead to vulnerabilities. Dependencies usually serve a purpose and it is generally a good idea to rely on components that are tried and tested, but dependencies also add to complexity and software that has significantly more dependencies than the competitors raise cause for concern. For all dependencies, we assume that source code is unavailable and analyse them using the techniques described in the following.

## No Source Code Available

When no source code is available, it is still possible to learn something about the code complexity, by examining the overall size of the binaries and any external dependencies that the system may have. Another simple measure is the set of features that the system offers and remember that from a security perspective "less is more", i.e. the number of necessary features divided by the number of total features should be as close to one as possible. Finally, it is possible to learn from history, by examining the number and severity of vulnerabilities previously discovered in the system.

The security history of a software system or service, a.k.a. security track record, includes all known vulnerabilities in the system and estimations of their severity.

Known vulnerabilities in third party systems are found in the Common Vulnerabilities and Exposures (CVE) database, which is maintained by the MITRE Corporation since 1999 [4]. It defines a common identifier (the CVE number) and contains a short description of the vulnerability. This provides security professionals an unambiguous way to discuss vulnerabilities, but it also allows anybody to search for all known vulnerabilities of a particular system. It is important to note that exercising responsible disclosure means that the system developer has patched most of these vulnerabilities. In addition to the total number of vulnerabilities found in a system, the CVE number includes the year that the vulnerability was first discovered, which may be extrapolated to provide a rough estimate of the vulnerability discovery rate of the system. The annual number of CVE numbers that mention one of the four big browsers over the past decade are shown in Figure 1.

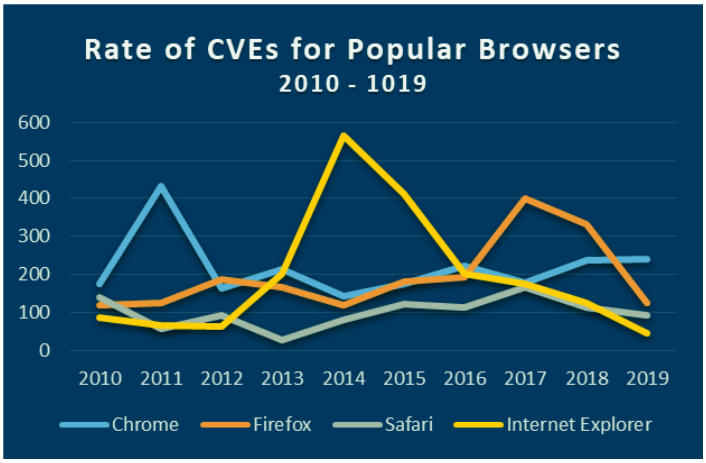


*Figure 1 CVE report on popular browsers*

The rate of CVE numbers, shown in Figure 1, indicate that there may be significant variations in the annual number of reported vulnerabilities for individual years. However, each product has a stable base rate of vulnerabilities per year, e.g. Chrome has around 200 reported vulnerabilities per year and Safari about half that number. The security of Safari may not be better than the other browsers, because the difference may be explained by Safari running on a Unix-based system, which makes it inherently more secure than software running on other systems. It should also be mentioned that a new version of Internet Explorer (IE11) was introduced in 2013, which partially explains the dramatic increase in the number of reported vulnerabilities after 2012.

The vulnerability spikes for Chrome in 2011 or Firefox in 2017-2018, cannot be explained by similar major version changes, but may still reflect fundamental changes to the underlying technology.

In addition to naming and describing known vulnerabilities through the CVE, the Common Vulnerability Scoring System (CVSS) [5] defines a method to capture the principal characteristics of a vulnerability and calculate a numerical score reflecting its severity. The severity measures how much control of the system an attacker may obtain through the vulnerability and how easy the vulnerability is to exploit in practice. The severity calculation includes no information about the value of the assets that the system protects, so it cannot replace a thorough risk evaluation.

Moreover, CVSS scores may change when more information becomes available, e.g. if the vulnerability becomes easier to exploit or is shown to have higher impact. The CVSS scoring system is maintained by a special interest group (SIG) of the Forum of Incident Response and Security Teams (FIRST) and CVSS scores for most vulnerabilities listed in the CVE database can be found on the National Vulnerability Database [6].

The CVSS scores vulnerabilities from 0 to 10, where 10 is the most critical. Numerical scores are, however, difficult to communicate to a large audience, so FIRST has decided to define severity levels that map the CVSS scores to text. This mapping is shown in Table 1.

| CVSS Score | Severity Level |
|---|---|
| 0.0 | None |
| 0.1 - 3.9 | Low |
| 4.0 - 6.9 | Medium |
| 7.0 - 8.9 | High |
| 9.0 - 10 | Critical |

*Table 1 Mapping CVSS Scores to Severity*

It is, however, important to understand the limitations of the CVE and CVSS scores. A high number of CVEs for a product does suggest that the product is insecure, but it may equally well indicate that the product is more popular and has been ported to more platforms, as is the case when we compare the number of CVEs for Safari and Chrome reported in Figure 1.

Similarly, the absence of CVEs for a specific product does not guarantee a secure product, because vulnerabilities may be found in one of the product's dependencies. For example, the CVE database does not contain an entry for the Ukrainian M.E.Doc accounting package, which was initially exploited in the Not-Petya ransomware attack that hit Maersk in 2017, but it includes both of the underlying vulnerabilities (CVE-2017-0144 and CVE-2017-0145).

Another important factor to consider is the way we currently release software to customers before it has been completely tested, which is one of the things that the certification scheme in the EU Cybersecurity Act aims to rectify. This means first versions of software contain more errors and more vulnerabilities, which are then removed as the software matures. The dramatic increase in vulnerabilities in Internet Explorer after Microsoft introduced IE11 in 2013, as shown on Figure 1, illustrates this point. This suggests that it may be more important to consider the trend in the number of CVEs, rather than the total.

One way to combine the CVE and CVSS scores for individual software systems is implemented in CodeTrust [7], which calculates a security score as the product of aggregated vulnerability scores and severity scores.

The aggregated vulnerability score considers both the total number of CVEs and the current trend (whether the CVE rate is increasing, stable or reducing). Similarly, the aggregated severity score considers the distribution of vulnerabilities across the different severity levels.

## Common Vulnerability Assessment Standards

A brief summary of common standards for vulnerability assessment are outlined here.

**CVE** (Common Vulnerabilities and Exposures) [4]
The CVE provides a database of vulnerabilities, where each new vulnerability is assigned a unique identifier, known as the CVE number or CVE Identifier. This facilitates information sharing and discussion of particular vulnerabilities. The CVE database contains brief descriptions of most publicly known vulnerabilities and exposures.

**CVSS** (Common Vulnerability Scoring System) [5]
The CVSS provides a common metric to rate the severity of known vulnerabilities, based on how easy it is to exploit (exploitability), what systems are affected by the vulnerability (scope) and the possible consequences of a successful attack (impact). It facilitates prioritisation of remedial efforts when a system is affected by several CVEs.

**CWE** (Common Weakness Enumeration) [9]
The CWE provides a classification (a structured list) of clearly defined software and hardware weaknesses, which is useful to describe the issues found in code reviews. Where the CVE describes the actual vulnerabilities in software and services, the CWE describes the underlying problems causing vulnerabilities.

**CAPEC** (Common Attack Pattern Enumeration and Classification) [10]
The CAPEC provides a classification (a structured list) of common ways to attack systems and services. Where the other vulnerability assessment standards focus on the software artefacts of the systems and services, the CAPEC focus on the system in operation, e.g. it provides identifiers and definitions of the different types of phishing operations.

The CVE and CVSS provide a comprehensive list of known vulnerabilities and their impact on the security of systems and services, where the CWE and CAPEC identifies the theoretical and methodological errors that may arise in working systems

## Company Security Assessment

Some software requires frequent patches and security updates, which reflects the competence, development methods and priorities of the development team, i.e. some development teams seem to favour feature rich software delivered at a high rate, while others favour more judicious methods focusing on formal specification, evaluation and testing. It is hard to know the development methodology of a software product that has not been subjected to the certification process. We therefore need to rely on inference based on externally observable indicators, such as the number of CVEs and the CVSS scores mentioned above.

## Security Team Assessment

The quality of software developed by a development team depends on the project management methodologies, processes and tools employed by the team, but also on the security consciousness and experience of the individual team members. Most of the development team, however, is unknown to the public outside of the open source software community, where modifications can often be traced back to individual developer. This leads some organisations to demonstrate their commitment to security by purchasing security companies with a high profile (and a healthy product portfolio) as British Telecom did when they purchased Bruce Schneier's company Counterpane in 2006. Another way to demonstrate commitment to security is to hire well known and respected security professionals to leading security positions, as Yahoo did when they appointed Alex Stamos as CISO in 2014, which he quit in 2015 because of a disagreement over a programme that would scan the emails of all Yahoo subscribers. Security profiles like Alex Stamos, however, are often conscious about the value of maintaining their reputation, so when Stamos joined Facebook as CISO in 2015, some people suggested that he would function as a human canary and that Facebook users should pay attention if he ever decided to leave [8].

It is difficult to quantify the security value of the individual team members, but when high profile security professionals decide to join or leave an organisation, it must raise concern. Moreover, if a company has a churn rate in their software development teams that is significantly above the industry average, it may suggest a problematic work culture and it definitely indicates a challenge to maintain the development team's corporate memory.

## Security Track Record

The security track record of a company aggregates the security track records of all the products and services that the company offers, i.e. it may be established through a combination of the common vulnerability assessment techniques mentioned above. The results of a search in the CVE database for the name of some well-known software companies are shown in Figure 2.

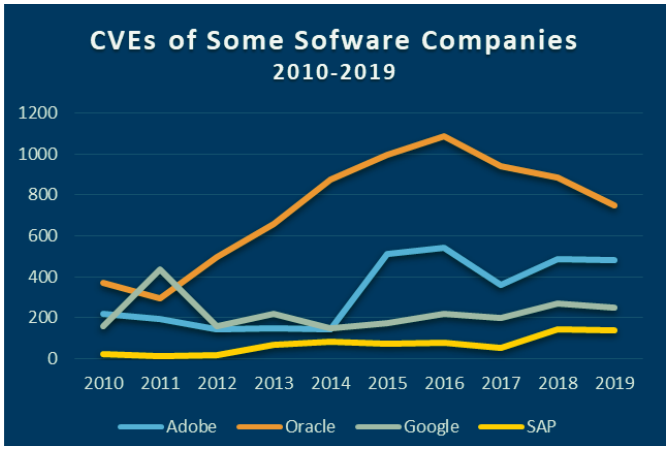


*Figure 2 Total number of CVEs for well-known companies*

One must be extremely careful when using CVE information to estimate an organisation's security posture, because larger companies with a broader and more exposed product portfolio will usually have more reported CVEs against them. Moreover, CVEs may relate to a few specific products, e.g. more than 90% of the CVEs reported for Adobe in 2015 relate to only two products Adobe Flash and Adobe Reader/ Acrobat. This is probably more an indication of the inherent difficulty in writing secure Internet facing interpreters, than any lack of commitment to security by Adobe.

We first observe that the total number of reported CVEs per year for these companies have almost doubled over the past decade (a little less for Google and a lot more for SAP). This may simply indicate that more security experts are examining the different systems, but it may equally well indicate a general increase in system complexity with the embrace of Cloud Computing and service outsourcing.

We also observe that Google has relatively few reported CVEs in view of the company's size and presence on the Internet, which we attribute to a healthy security culture at Google. This indicates that considering the total number of CVEs reported for a company, and its trend, provides useful information about the company's security posture, albeit one that must be used carefully.

## Summary

In this Part 2 of the whitepaper on cybersecurity certifications, we examined ways to assess the security of software that has not yet been certified. This assessment must therefore be based on simple metrics based on public available information. We identify the following five simple steps for security assessment of third party software and services.

**Step 1.** Determine the complexity of the system under consideration, e.g. using some of the complexity metrics presented here.

**Step 2.** Establish the security track record of the system provider by aggregating open source intelligence data, such as CVEs and CVSS scores for the software; remember to consider both the current status and trends.

**Step 3.** Establish the security track record of the system provider through aggregating the open source intelligence data about all the provider's products and services.

**Step 4.** Evaluate the provider's security team. Experienced security professionals know their value and gravitate towards companies with a healthy security culture.

**Step 5.** Consult your network. There is limited openness about security incidents in the industry, so there is no basis for a Trustpilot of cybersecurity, but there is often a willingness among security professionals to discuss problems and experiences with third party software or services with colleagues and friends from other companies.

Finally, it is important to remember that the metrics and assessment methods presented in this whitepaper only provides rough estimates. The results should not be interpreted on an absolute scale, but are generally better suited to compare similar products from competing companies.

# References


1. T. J. McCabe, "A Complexity Measure," IEEE Transactions on Software Engineering , Vols. SE-2, no. 4, pp. 308-320, 1976.
2. M. H. Halstead, Elements of Software Science (Operating and Programming Systems Series), {New York, NY, USA: Elsevier Science Inc., 1977.
3. J. Fitzpatrick, "Applying the ABC Metric to C, C++, and Java," in More C++ Gems, New York, NY, USA, Cambridge University Press, 2000, pp. 245--264.
4. MITRE Corporation, "Common Vulnerabilities and Exposures," Mitre Corporation, [Online]. Available: https://cve.mitre.org/index.html.
5. Forum of Incident Response and Security Teams, "Common Vulnerability Scoring System SIG," [Online]. Available: https://www.first.org/cvss/.
6. Information Technology Laboratory (ITL), "National Vulnerability Database," National Institute of Standards and Technology, [Online]. Available: https://nvd.nist.gov/.
7. C. D. Jensen and M. B. Nielsen, "CodeTrust: Trusting Software Systems," in Proceedings of the 12th IFIP WG 11.11 International Conference on Trust Management, Toronto, Canada, 2018.
8. J. Menn, Cult of the Dead Cow: How the Original Hacking Supergroup Might Just Save the World, New York, NY, USA: Public Affairs, 2019.
9. Mitre Corporation, "Common Weakness Enumeration," Mitre Corporation, [Online]. Available: https://cwe.mitre.org/.
10. Mitre Corporation, "Common Attack Pattern Enumeration and Classification," Mitre Corporation, [Online]. Available: https://capec.mitre.org/.

DTU

Technical University of Denmark

DTU Compute Master in Cyber Security Whitepapers

This Master in Cyber Security whitepaper is part of a series of whitepapers intended for alumni of the Master in Cyber Security Programme at DTU and security professionals working in Danish companies, with special interests in the topics covered in the whitepaper, or security professionals who may simply wish to broaden their horizon. There is no fixed schedule for these whitepapers, but we expect to publish 4 – 6 whitepapers annually.

Topics of these whitepapers may cover all aspects of cyber security, from user awareness and insider threats to advanced technologies and emerging security paradigms, but the main focus of this series will be on the practical application of theoretical and technological advances in cyber security.

The Master in Cyber Security whitepaper series is published by:

Department of Applied Mathematics and Computer Science
Technical University of Denmark
Building 324
Richard Petersens Plads
DK-2800 Kgs. Lyngby
Denmark