National University of Public Service
Doctoral School of Military Engineering

Gergely Mészáros

Systematic security analysis of Open Source and Free Software used in Critical Information Infrastructures

Author's summary of the PhD Dissertation

Supervisors:

Col. Prof. Dr. Zsolt Haig, PhD **Lt. Col. Dr. Lajos Muha, PhD**

Budapest, 2020

Scientific problem

Today's information society based on many complex and intertwined technologies, poses an increasingly serious challenge to security professionals. The complexity of systems, the number of tools and methods used, is constantly increasing, slowly reaching a level that no human mind is able to absorb. It's enough to think about training. While a few decades ago there was the IT training (even electrical engineer training), today there are many specializations like cloud technology, data analysis, artificial intelligence research or cybersecurity which could be further broken up to cryptography, network security, operations and other risk management. These are becoming separate professions nowadays. Due to the exponential growth of knowledge, it is now inconceivable for anyone to be an IT polyhistor.

Our complex world poses complex threats, and considering the omnipresence of information technology, we get a rather dangerous mixture. This is particularly dangerous in an area where human lives could be at stake, where a potential failure or attack could cause significant damage, i.e. in the field of Critical Infrastructures (CI).

Critical Infrastructures may have complex interdependence, therefore even if the system itself isn't considered to be critical, its failure could make another CI fail to function, causing the disruption spreads in time and space leading to serious problems in the supply chain, disturbing the economy or the government.

One of the most visible phenomena in the development of information technology is the increasing adoption of open source technologies. Few decades ago it was seen as an interesting hobby, perhaps even just a naive ideology, but for today it has grown into one of the most important factors in the industry, backed by a considerable number of business partners.

The concept of Open Source (FLOSS[1]) has come a long way in recent years since the initial, primarily ideologically motivated Free Software idea. This modern version, supported by market players, is clearly a mainstream technology, a commercially acceptable, even desirable element that is often used in both software development and distribution. The benefits of using it are obvious: it allows access to the source code, allows us to distribute derivative products, and makes cutting-edge technologies available without significant investment. It is difficult to compete with these potential benefits.

We are facing open source projects in almost every field, open development is considered innovative and "trendy". Major tech companies like Google, Facebook or even Microsoft are trying to convince us of their close friendship with Open Source.

The presence of Open Source is especially strong in the field of IoT[2] technologies, web servers and server market in general. The Apache™ Foundation, which was organized around a single - though very popular - web server, manages nearly two hundred different projects today. Many of these projects are not insignificant at all. For example, the Apache™ Hadoop® market, projected to exceed $50 billion by 2022. The

[1] Free Libre and Open Source software.

[2] the "Internet of Things" is a complex network of physical devices (buildings, vehicles, household and surveillance, etc.) that can communicate with each other usually in a wireless way.

Open Source based browsers dominate the market, what is a significant change compared to a few decades ago. Open Source technologies are prevalent in both the business world and research.

The proliferation of business backed Open Source helped the classic model as well. The acceptance of Open Source as a viable alternative has greatly increased and FLOSS technologies implicitly or explicitly invade new areas previously dominated by business products exclusively. The extensive use of open source components - and thus Open Source - has an indirect effect on all areas of our technology.

Therefore the following questions arise: whether this change will have an impact on the security of information systems, are there any significant differences between FLOSS and business systems and components, and will it be necessary to change our currently applied security procedures.

The FLOSS development methodology is an attractive option for both technology giants and small start-ups. Due to its independence, openness and transparency, its use is also appealing in public administration.

Software and component transparency can bring significant benefits and may even become a requirement in the government over time. High availability and large selection greatly shortens development time, which simply cannot be ignored by the market. Thus, the use of FLOSS can also be a strategic goal. In my experience, however, organizations often do not have dedicated FLOSS-specific regulation. The development environment and application of FLOSS may differ from the norm to such an extent that the existing regulations are no longer suitable for its management. This can be equally true at the organizational, state, or international levels.

Of course, the whole issue can very easily be swept under the rug, by stating that the organization does not use any open source products, therefore the matter does not need to be addressed. However, this approach is, in my view, insufficient and even dangerous today. Your organization may not be aware of the FLOSS elements used. Due to the extremely widespread popularity of FLOSS components in development, suppliers and developers are likely to use FLOSS components, and the organization's employees may use such a product outside the system. For this reason, the issue of FLOSS should be addressed even if the organization completely avoids FLOSS, at least to the extent that this avoidance can be ensured in practice through regulation.

In summary, research of the security implications of FLOSS is motivated by three factors:

- increasing cyber threat;
- the increasing use and implicit nature of FLOSS usage;
- lack or inadequacy of existing FLOSS specific regulations.

The impact of the open model on information security cannot be analyzed without examining where and how the information system can interact with FLOSS systems. One of my study objectives was to include not only the "Free to Use" Free Software itself, but all facts and information that are publicly available about the Open Development model. In addition to Free Software, I took into consideration any FLOSS components, source repositories, software repositories, moreover all metadata created and used during

Open Development. Communication channels of the Community, social and economic conditions, and relationship of the actors should also be examined.

Given that exploitation of FLOSS can be realized in multiple ways, it is worth examining all cases. The organization can be a direct or indirect consumer through its suppliers, it can be a community participant or an actor producing its own or products using FLOSS as part of the development process. Important features are the possibility of direct participation in the development and component integration.

To the best of my knowledge, there is currently no research that analyzes the security implications of the Open Development Methodology in a complex way. Many studies address the issue of Open Source quality assurance, even more so the perceived or real advantages and disadvantages. However in order to understand the security implications of an Open Source development methodology, it is not enough to approach the issue from a single point of view.

The impact of FLOSS on security can be significant, multifaceted and hidden. From both a scientific and a practical point of view, the extent and far-reaching impact of this phenomenon on the security level of high priority systems, in particular Critical Information Infrastructures, is fascinating. Are existing regulations sufficient? Are organizations able to adapt to the changing circumstances and manage potentially unusual risks arising from the FLOSS usage?

During the planning of the research, I tried to find answers to these questions.

Research goals

The primary objective of the research was to determine the impact of the current advancement of FLOSS technologies on the security of information systems with high security requirements, in particular the Critical Information Infrastructures (CII). In order to answer the question posed in the objective, it was necessary to subdivide it to several parts.

On the one hand, it needs to be determined how the FLOSS and the Critical System Information System can be linked and what level of cooperation is necessary to ensure that the impact on safety can be measurable. On the other hand, we need to identify the FLOSS features that can have a specific security impact on your information system. After defining these features and impact points, the next step is to categorize and systematize the specific security effects and the operations in order to exploit or avoid them. Finally, in my opinion, the significance of the effects of the FLOSS phenomenon can be assessed by identifying areas already covered and not covered by the existing regulation, i.e. identifying the areas where FLOSS can actually have positive or negative effects. It therefore seemed necessary to draw conclusions about the actual safety effects of the Open Development Model by comparing the identified effects with the existing protection measures and regulations.

In addition to the above, in order to ensure the reproducibility of the research, my goal was to apply a well-defined and documented methodology and to make all research materials and data publicly available

according to the Open Science guidelines. In this way, hopefully, every step of the research can be tracked and the information gathered can be easily used by other researchers.

In accordance with the above, I formulated the following research objectives:

- Identify and systematize the features of the open model that may affect IT security.
  - Assess the extent to which effects of FLOSS have been researched in different areas.
  - Summarize FLOSS features in a comprehensive system.
  - Identify features that affect security.
- Identify the positive and negative impacts that may affect the safety of Critical Infrastructure.
- Propose measures to avoid negative effects and exploit positive effects.
- Compare the results with the current domestic regulations and draw conclusions about the applicability of FLOSS in CII.

Research Hypotheses

In accordance with the research goals, I set up the following research hypotheses:

H1. I assume that the open development model and its product have unique properties that affect the security of information systems in a special way.

H2. I assume that the risks arising from certain specific features have a direct or indirect impact on the security of Critical Information Infrastructure.

H3. I assume that it is possible to define protection measures that can mitigate the risk posed by security issues arising from the unique features of FLOSS usage or may provide an alternative solution to existing problems.

H4. I assume that a certain group of FLOSS products of different quality and source can be used in Critical Infrastructures, provided that appropriate new protection measures are put in place.

Research method

In line with the research objectives, I have tried to find a method that provides a comprehensive picture of the whole issue, but at the same time allows a systematic analysis to keep the degree of subjectivity as low as possible. The chosen methodology should therefore be systematic, reproducible and systematic.

In case of empirical analyzes, the method of data collection can be determined by using hierarchical question formulation. Between the levels using the method of deduction and induction, it is possible to formulate research questions from the generalized question then draw conclusions from the collected results. During

the research design phase, I've followed the methodology proposed by Punch, Leshem, and Trafford. I first set up the research questions, then the conceptual framework, finally I determined the methods and the structure of the research.

fig. 1 shows the conceptual framework used. Here one can read the necessary steps of the research, the sources of information processed during the research and the expected results.

The first column of the figure shows the information sources. The primary source of information for this research is data collected manually from processed articles and studies. During the impact analysis, I also took recommendations of existing, actively used frameworks (ISO / IEC 27001, NIST 800-53, Common Criteria) into consideration. The completeness formulated in the research objectives is thus ensured in terms of FLOSS features and safety effects.

From the steps of the research process (marked with rounded bricks in the second column), it can be seen that the collection-analysis part of the research is cyclical, i.e. the taxonomy constantly refined during the impact analysis and the expanding categories of safety impacts were reused during data collection. The cyclical implementation was necessary because the new categories created during the labeling had to be assigned to the already processed publications as well, and the new publications published had to be continuously integrated into the system.

The results in the third column indicate documents and databases that contain information and data that was used in later phases of the research, or can be used in other research and in practice.

To achieve the first research goal, I used the method of Systematic Mapping Study proposed by Petersen.

Systematic Mapping Study method is a widely used practical tool for classifying software engineering areas and assessing the structure of research state. This analysis focuses on the number of publications by category to determine the estimated coverage of the area by category. Classical SMS in contrast to the literature review does not emphasize details, therefore publications are usually not analyzed in detail. Compared to the proposed method, my case required a deeper analysis - approaching the classical systematic review - since I also wanted to determine the methodology of the publications used and the type of results.

For FLOSS feature categories, I used the FLOSS taxonomy sketch produced during the preliminary research, which I gradually refined with the help of the collected materials. The collection work that formed the basis of the SMS served a dual purpose: firstly, the SMS made it possible to determine to what extent the research was carried out, i. e. the expected level of reliability and completeness of the results in each area. Secondly, the collected and labeled source material could form the basis of the next analytical phase.

First phase of mapping and data collection was carried out in 2016. It took a considerable time to process and analyze the entire collected material, so I constantly updated the existing publication database. However, the statistical data of the SMS has become obsolete, so I repeated the categorization in 2020 using the same search methodology. Literature research thus reflects the state of the beginning of 2020.

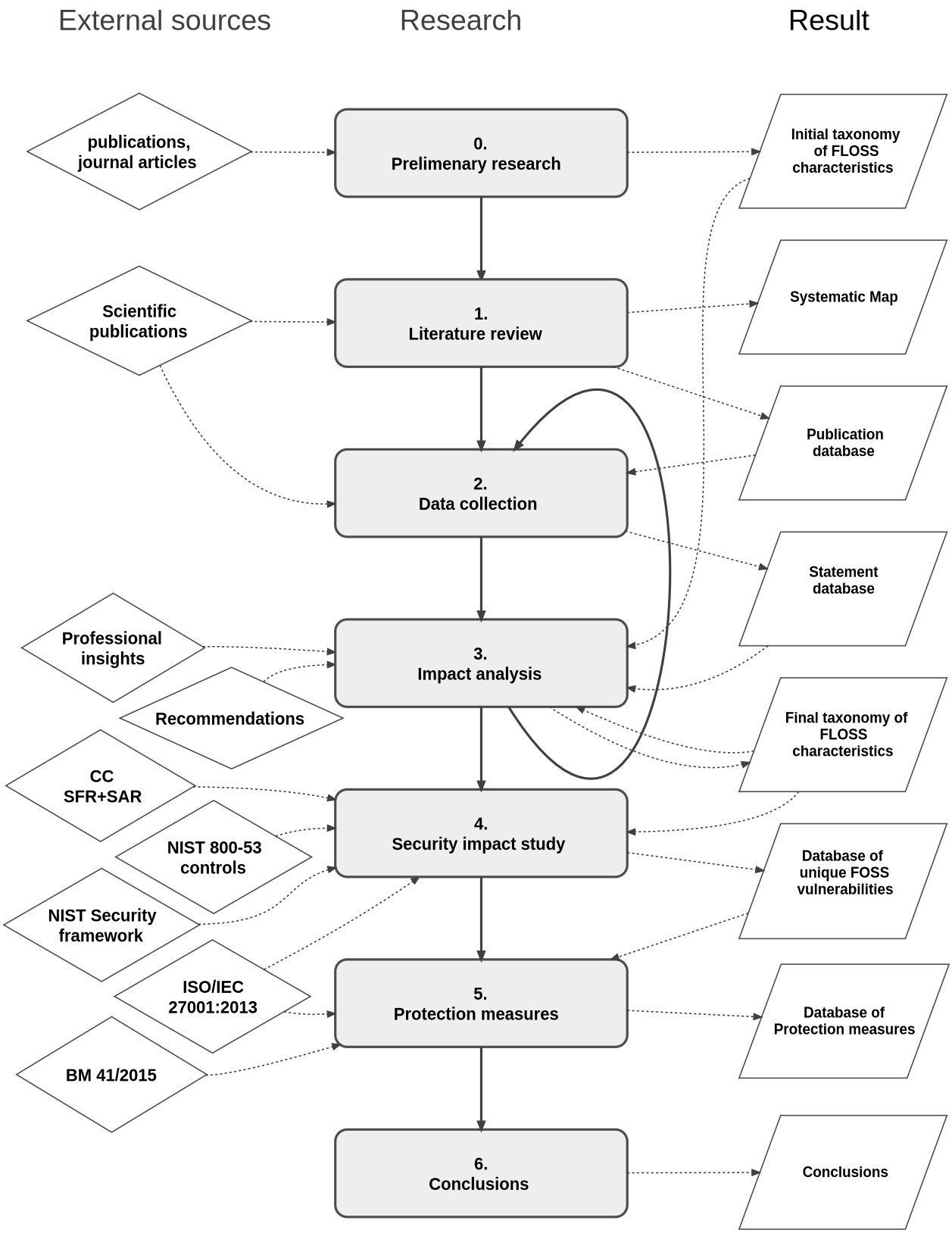

Figure 1: Conceptional framework

In order to achieve the second research goal, analysis-synthesis methodology was applied. From the documents collected and continuously updated in the first phase, I built the most complete possible model of the features of FLOSS, then based on the model I extracted and proposed solutions and characteristics related to the safety effects. By synthesizing the output of the first research phase, I determined the areas and points according to the second research goal of the framework, and I provided an estimate of the number of unidentified features and protection measures.

Comparing the model of FLOSS features and effects, as well as the recommendations and regulations for high security systems, I determined the possible protection measures formulated under the third goal. The development of protection measures was inspired by the NIST 800-53 security overlay mechanism. Following the idea, I planned to create a set of measures for Open Source using highly critical organizations, especially Critical Infrastructures, which, in addition to the existing regulations, would draw attention to points that need to be addressed differently or with special attention in FLOSS.

Following similar vision, I planned to create a set of measures for critical open source user organizations, in particular for Critical Infrastructure, which, in addition to the existing regulations, would draw attention to instances that need to be treated differently or with special attention in the case of FLOSS.

Finally, I compared the identified effects and protection measures with the relevant recommendations and domestic regulations in order to reveal possible blind spots.

Structure of the Dissertation

The dissertation consists of six chapters. The first chapter focuses on defining research objectives and methodology. The second chapter deals with the subject and literature of the research. Here I define the framework of the research topic and the studied areas. It also contains the summary results of the literature search, which, in accordance with the research objectives, are intended to determine the thoroughness of the examination of each safety area, on the basis of which the completeness of the results can be estimated. At the end of the second chapter, I examine the interfaces between Critical Infrastructure and open source, which serve as the basis for the security analysis discussed in the following chapters.

Internal and external unique characteristics of Open Source were split into two separate chapters in order to maintain a balance. The third chapter examines internal properties of FLOSS, while the fourth chapter analyzes the features arising from the relationship between external factors and FLOSS. The chapters summarize the specific features defined in Objective 1. Based on these, the impacts and proposals mentioned in Objective 2 can be identified.

The taxonomy that serves as the guiding principle of the third and fourth chapters is shown in fig. 2. All chapters have been coded for easy identification in the form of FS-category-subcategory.

For each analyzed feature categories, I identified the vulnerabilities as well as suggestions. Vulnerabilities and suggestions were summarized in tabular form at the end of the subsections.

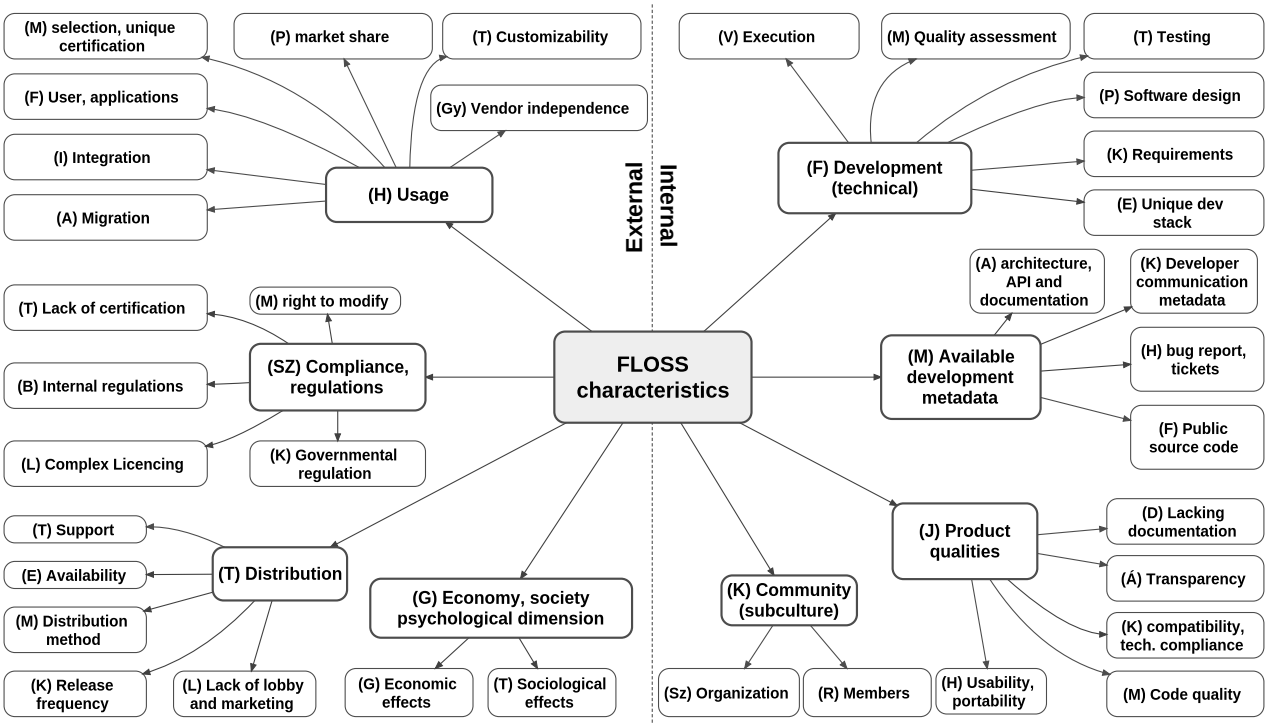

Figure 2: Taxonomoy of Open Source characteristics

The following sections analyze the impact of FLOSS features in Critical Infrastructures. This is done in two steps. In the fourth chapter, I examine possible effects of Open Source features along the information security guidelines defined by the NIST Cyber Security Framework, then, in the fifth chapter I compare the identified protection measures with local regulations governing the security of Critical Infrastructures.

For clear identification, features, measures and security guidelines have been clearly coded, which I have compiled in a relational database for easy processing.

Summary conclusions

In my dissertation, I examined the usability possibilities of software components derived from the Open Development Model (hereinafter FLOSS) in Critical Information Infrastructures (hereinafter CII).

Applying systematic analysis on available publications, I found that the scientific community is significantly more interested in the FLOSS development methodology, while other aspects (safety, compliance, life cycle management) are less researched. The results, which also address security issues are largely limited to software quality testing of Open Source products. Due to the easily accessible source code, FLOSS products are often used as research data sources, but the specific implications of openness are not usually addressed in publications.

The proposed methods show even distribution in terms of features. It seems the research community considers all features to be equally important. At the same time, empirical data and basic research re-

sults are available primarily in the field of Open Source development, as the majority of the results are concentrated here. I found that the data needed to examine some characteristics like economic impact, psychological dimension, and product properties can currently be collected only from surveys focused on a (non-representative) narrow population or there are no empirical results available at all.

I found that the use of open source is not always obvious, the components and subsystems used can be incorporated in multiple indirect ways. The clear negative impact of multiple indirect uses on security is that vulnerabilities are fixed much more slowly, zero-day vulnerabilities may remain exploitable for long periods of time. *Consequently, it is not recommended to completely exclude Open Source analysis from the CII IT security regulations even if such a system is not (apparently) used by the organization.*

I have shown that the connection between CII and FLOSS can be realized in four main ways: acquired as business product purchased from a reseller or as part of it, obtained directly from the community in binary or unmodified executable form, compiled internally using modified / verified source code; finally via continuous integration as part of the development community, with active participation in development.

At higher levels of connection and collaboration, new protection options unknown to closed source open up, but at the same time, the number of potential vulnerabilities increase.

Applying systematic analysis on Open Source characteristics, I found that FLOSS and its development methodology have unique characteristics in certain areas that can affect IT security in both positive and negative ways. I have discovered that for some of the problems, the research community already has suggested solutions, but many issues remain open and often unregulated.

Considering its development model, the FLOSS methodology appears to be very special, with features that would be a critical shortcoming in closed development, yet through extensive testing and alternative code acceptance methods, it is ultimately able to produce good quality code. I have found that the two models have been steadily converging in recent years, both borrowing certain elements from the other, but the differences are still significant enough to pose difficulties and thus security risks during cooperation.

FLOSS software and components do not differ significantly from the business variants in terms of product features, both models can deliver exceptionally high quality and poor products. The main differences are the (non)existence of quality certification and the verifiability of the certification. FLOSS products are usually more portable but direct safety effects of portability have not been identified.

The organizational and social structure of the open and closed models differ significantly. The structure of the open development community resembles a community network, its management requires high technical training and good social skills. It is not an easy task to influence the open community, but its operation is completely transparent, so the potential risks can be estimated more easily. Effectiveness is usually a prerequisite for long-term existence of the community, therefore participating, managing, but at least analysing the community should play some role in supply chain risk management tasks. In case of strategic usage, close cooperation, which can be achieved through participation, is essential.

With regard to metadata, the obvious difference is openness. The source code, system API documentation,

system design, bug reports, responses, and, in many cases, even bug fixes are openly available. This transparency offers unique opportunities but also carries security risks. Based on the metadata, qualitative or even quantitative metrics can be used to characterize the quality, readiness, response time and general condition of the project and its community. It is often possible to identify prominent developers and even the identity of the authors of different code fragments. In the same cases, operational correctness of the entire source code can be formally verified. In contrast, sensitive information can become public through bug report tickets or source code. Via source code and error ticket analysis, vulnerabilities can be detected and exploited before the fix, and all users can be attacked by hidden malicious source code manipulation. The process of bug reporting must be approached differently than in the case of business applications, incorrectly reported error tickets may delay or even suspend bug fixing indefinitely.

Open source phenomenon has significant economic and social benefits. Unfortunately, consequences of FLOSS and the open model are rarely part of the curriculum in most educational institutions, although there has been a positive change in recent years. In order to strike the right balance, special attention should be paid to FLOSS-specific education. In terms of economic and social impact, I have not been able to identify direct security implications, apart from the fact that knowledge of the phenomenon is a prerequisite for the application of proper protection procedures.

In case of FLOSS, selection of the right product is problematic, the usual qualification systems cannot be used, unique databases and models are needed. The very extensive FLOSS component integration faces specific problems. Particular attention should be paid to managing local change. The difference between attitudes of the management and software engineers towards FLOSS can lead to the covert use of FLOSS components, which can lead to serious safety issues in the absence of regulation.

I have found that the regulation of FLOSS is currently not fully mature. In legislative terms, initial steps have been taken, but at company level there is generally no explicit FLOSS-specific regulation, it may not address the issue at all, and certification schemes adopted do not fit well with FLOSS. FLOSS licensing is complex, and while it does not have immediate security implications, still can compromise availability. Licensing is not always clear, and due to the typically component-based structure, situation assessment can also be a challenge. Due to the shortcomings identified, in particular the lack of certifications, higher levels of cooperation is very important, as manufacturers – in this case communities – might not be able to obtain the necessary certifications or generally not interested in that. Certification is usually important in high security applications, but it is extremely resource-intensive to perform individually, which significantly hampers the usability of the FLOSS product.

In the absence of business support, the distribution method of the open model shows clear differences that can also have some impact on security. Community support is unreliable, difficult to plan in time, requires social skills, and market success does not affect strongly the support quality. In principle, integrity of binary products or components can be verified on the basis of the source-code, but in practice this is still difficult to achieve, so trust in the third party packaging actor is still a requirement. Organized training, which is the basis of internal support, is often lacking, making it difficult to find professionals with proper

knowledge. In the absence of a formal training framework, the expertise can only be verified by admission tests.

Based on the above, I concluded that the products and components derived from the Open Source Development Model have unique characteristics that may affect the level of IT security of the user organization. **Based on the results I accepted hypothesis H1.**

I have shown that Open Source components can have a covert effect on Critical Information Infrastructures through multiple, possibly implicit channels, FLOSS characteristics may imply risks that directly or indirectly affect the security of CII through these channels. **Based on these results, I accepted hypothesis H2.**

Using methodology proposed in the NIST Cybersecurity Framework (which is specifically tailored for CII analyzes), I have designed protection measures to mitigate the risk posed by FLOSS-specific vulnerabilities. Open Source specific elements were identified in all major categories, consequently *the Open Source regulation should cover the entire security process*.

I have found that FLOSS exploitation above COTS (off-the-shelf) usage *in contrast to best practice for business products, there are specific opportunities and needs that need to be addressed in identification, protection, discovery, response and recovery planning.*

Some protection measures simply aim to reduce risk arising from vulnerabilities special to FLOSS, but in some cases new opportunities have been identified that are unaccomplishable in case of closed source products.

Based on the defined protection measures, I accepted hypothesis H3.

Comparing FLOSS vulnerabilities, recommendations, protection measures and legal requirements, I found that use of FLOSS has clear advantages for the protection of CII, however the highest level of security requirements cannot be fully met.

The local policy expects the organization to contract out information security roles and responsibilities for all external partners. In the case of open communities, this is certainly problematic. Verifying identities of the contractor is also dubious, considering the anonymous nature of the open community. The FLOSS development community, and the software in general, primarily use graph-like WoT-based [3] cryptographic certificates instead of hierarchical PKIs, which is not addressed by the legislation. If we consider PKI certificates to be mandatory, we exclude a significant range of FLOSS systems as well as their internal functionality like automated updates, without which other requirements become unfeasible. Due to transitivity, all security regulations also apply to the developer, therefore the open community should ultimately have to meet all the requirements, which in some areas (e.g. classified source-code, administrative measures) will most certainly not be met. The law stipulates that the development documentation of the information system must not be accessible to and modified by unauthorized persons, which is virtually

[3] Web of Trust, an alternative to the better-known PKI framework. Can be used without a central trusted partner.

impossible in case of active participation in the open development model. FLOSS projects very rarely obtain certification that is legally recognized, so they cannot be applied in cases where certification is a prerequisite. The qualified developer should have a security plan prepared and document the process of implementation set out in the plan. This step is missing in the vast majority of open source projects.

In the current regulatory environment, the use of FLOSS is cumbersome, and at higher security levels, working closely with the community is downright impractical. Exploiting the benefits and some of the protection measures require active involvement, which would often be overwhelming for the organization.

Based on the above, it can be stated that the direct use of open projects violates the regulation, classic FLOSS products do not fully meet the requirements defined by the highest level of security, therefore they can only be used in third-party products under the current regulations. **Hypothesis H4 must therefore be rejected.**

New Scientific Results

1. Using a systematic processing method, I defined and classified special characteristics of Open Source softwares and components.

To understand the impact of Open Source on Critical Information Infrastructures, it is first necessary to clarify the differences and characteristics compared to products derived from traditional software development. The security impact of Open Source features is not necessarily clear, so it is not enough to address certain effects selected on an ad-hoc basis. Therefore, a comprehensive feature database had to be created, on the basis of which the systematic analysis could be performed.

2. Using an analytic method performed along the new classification, specific impact of the Open Source characteristics on Critical Information Infrastructures at different security levels has been determined, and possible impact mechanisms have been identified.

In order to identify the specific security impacts, it was necessary to identify the possible impact vectors of the Open Source Development Model through which the impact in question could occur. After understanding the mechanism of action, a database was created by organizing the effects identified during the analysis of the characteristics and the mitigations proposed by the research community, which stores the available safety-related information in a uniform, filterable and searchable way.

3. By comparing the unique FLOSS characteristics and the categories of the NIST Cybersecurity Framework suitable for CII qualification, I have defined possible protection measures that can mitigate risks relevant to FLOSS-using organizations.

The solutions proposed by the research community do not always adequately cover the problems identified, and even if there is a proposed solution, finding the information can be extremely time consuming. It is therefore a logical step to create a single protection measure database that records the measures in a search-

able and systematic way. These can be used to reduce the impact of known vulnerabilities. The database should take into account unique characteristics of CII and cover as large a portion of IT security as possible. To this end, I followed the categories and recommendations of the CII-oriented NIST Cybersecurity Framework when developing protection measures.

4. Comparing the domestic legal environment regulating the safety requirements of Critical Infrastructures, the unique characteristics of the Open Source Model and the defined protection measures, I discovered that classic FLOSS products do not fully meet the requirements defined by the highest security level, therefore may only be used when incorporated into third party products.

Defining security measures and characteristics alone is not yet sufficient to ensure the protection of CII using FLOSS components. Certain protection measures may be impracticable and may conflict with expectations and guidelines tailored for closed source. However, comparing impacts and measures with the domestic regulations, the critical elements that currently hinder or make it impossible to use FLOSS can be identified. Based on my results, these issues can be clarified at the organizational or global regulatory level.

Recommendations

The research results summarized in my dissertation can be used to design security measures for any Critical Information Infrastructures where Open Source systems are used, will be used, or will be integrated in some way, such as through suppliers or under an SLA agreement. In my opinion, this is inevitable in the long run, so the results presented here can provide useful help to the organization's security professionals during preliminary calculations and training, and provide supplementary information for the organization's information security policies.

The Nist Security Framework, the ISO 27000 series and the Hungarian legal system (Decree BM 41/2015. (VII. 15.)) all require regular security awareness training. This training may only be performed by a qualified person. The dissertation uses a number of standards and recommendations, so its direct use for educational purposes is not recommended, however, I recommend my results to VET organizations and professionals as a source material and some more easily interpretable parts as a basis for teaching aid.

Future research may aim to determine the severity of individual FLOSS-specific vulnerabilities and assign them to attack probabilities. The proposed protection measures in their current form can only be applied on ad-hoc basis, as their need should be identified in the pre-regulatory risk assessment, which cannot be carried out with adequate precision in the absence of attack probabilities and severity metrics.

As risk management is difficult to generalize, it would be useful to establish a methodology to estimate these values for the problems identified. Given the potential risks and probabilities, the system presented in the research could form the basis of an Open Source Risk Assessment expert system, for which there is a growing demand in parallel with the increasing popularity of Open Source.

Publications of the author

Peer reviewed journal publications in Hungarian

- Mészáros Gergely: Nyílt fejlesztői közösségek hatása az informatikai biztonságra, Hadmérnök vol. XV.: issue 3. (in press) (2020)
- Mészáros, Gergely: Nyílt forráskódú rendszerek biztonsági kérdései, BOLYAI SZEMLE XXII : 1 pp. 63-76. , 14 p. (2013)
- Mészáros, Gergely: Elosztott verziókezelés a közigazgatásban, Hadmérnök IX : 3 pp. 191-206. , 16 p. (2014)
- Mészáros, Gergely: Információs rendszerek fenyegetéseinek képesség alapú osztályozása, Társadalom és Honvédelem XVII : 3-4. pp. 215-227. , 13 p. (2013)
- Mészáros, Gergely: Szun-Ce elvei a digitális világban, Hadmérnök VIII. : 2 pp. 377-388. , 12 p. (2013)
- Mészáros, Gergely; Hufnagel, Levente: Filtering outliers in OMTK Data Set, ANNUAL NEWS OF THE SZENT ISTVÁN UNIVERSITY YBL MIKLÓS FACULTACY OF BUILDING SCIENCES 7 : 1 pp. 42-46. , 5 p. (2007)
- Mészáros, Gergely: Magyarországi szántóföldi tartamkísérletek adatainak mesterséges intelligencia alapú elemzési lehetőségei Tudományos Közlemények Szent István Egyetem Ybl Miklós Műszaki Főiskolai Kar 1 pp.32-34 (2004)

Peer reviewed journal publications in English

- Mészáros, Gergely: Auditing Community Software Development, YBL Journal of Built Environment 3 : 1-2 pp. 26-33. , 8 p. (2015)
- Mészáros, Gergely: Yield Prediction Based on Neural Networks, Annual News of The Szent István University YBL Miklós Faculty of Building Sciences 3: pp. 90-95., 6 p. (2005)
- Gergely: Human-Computer Interaction through Hand Gestures, Annual News of the Szent István University Ybl Miklós Faculty of Building Sciences pp.38-41., 4 p. (2003)

Peer reviewed conference publications in English

- Mészáros, Gergely: Lessons of Transparent Collaboration: Comparison of E-Government and Software Developer Communities, In: Balthasar, Alexander; Golob, Blaž; Hansen, Hendrik; Müller-Török, Robert; Nemeslaki, András; Pichler, Johannes; Prosser, Alexander (szerk.) Central and Eastern European e|Dem and e|Gov Days 2016 : Multi-Level (e)Governance : is ICT a means to enhance transparency and democracy? Vienna, Ausztria : Austrian Computer Society, (2016) pp. 383-392. , 9 p.

- Mészáros, Gergely: Security impacts of community based software development pp. 325-336. In: Alexander, Balthasar; Blaž, Golob; Hendrik, Hansen; Balázs, Kőnig; Robert, Müller-Török; Alexander, Prosser (szerk.) CEE e|Dem and e|Gov Days 2015 : Time for a European Internet? Wien, Ausztria : Austrian Computer Society, (2015) p. 629
- Mészáros, Gergely: Pattern Classification via Neural Networks, In: Bergmeister, K (szerk.) Proceedings of the 3rd International PhD Symposium in Civil Engineering : Vol.2 Wien, Ausztria : Fleck Druck Gmbh, (2000) pp. 475-478. , 4 p.

Conference presentations

- Mészáros, Gergely: Katasztrófavédelem és nyílt forrás, In: Kiss, Dávid; Orbók, Ákos (szerk.) A haza szolgálatában 2014 konferencia rezümékötet, Budapest, Magyarország : Nemzeti Közszolgálati Egyetem, (2014) pp. 66-68. , 3 p.
- Mészáros, Gergely: Nyílt forráskód létjogosultsága a kormányzati rendszerekben, In: Keresztes, Gábor (szerk.) Tavaszi Szél, 2013 : Spring wind, 2013. 1-2. kötet, Budapest, Magyarország : Doktoranduszok Országos Szövetsége, (2013) pp. 46-54. , 9 p.
- Mészáros, Gergely: Kritikus infrastruktúrákban felhasznált nyílt forráskódú rendszerek auditálási kérdései, In: Szakál, Béla (szerk.) Intézeti tudományos konferencia, Budapest, Magyarország : Avernim, (2012) pp. 91-97. , 7 p.
- Mészáros, Gergely: GIS rendszertervezés nyílt forráskódú alapokon, In: Márkus, Béla (szerk.) GISopen 2011 : Megfelelni az új kihívásoknak Székesfehérvár, Magyarország : Nyugat-magyarországi Egyetem Geoinformatikai Kar, (2011) pp. 55-64. , 10 p.

Conference presentations in English

- Mészáros, Gergely: IoT security and education, In: Talata, István (szerk.) Matematikát, Fizikát és Informatikát Oktatók 41. Országos Konferenciája : MAFIOK 2017, Budapest, Magyarország : Szent István Egyetem Ybl Miklós Építéstudományi Kar, (2017) pp. 1-7., 7 p.

Professional and scientific curriculum vitae

Name:	Mészáros Gergely Tibor
Birth:	Budapest, 1975 május 24.
Nationality:	Hungarian

Education

2012-2015 **Doctoral School of Military Engineering**; Attack and defense of electronic systems, NUPS
1993-1999 **MSc in Surveying and Geoinformatics**; University of Technical University of Budapest
1989-1993 **Graduation**: Babits Mihály Gimnázium, Budapest faculty of mathematics and physics
1998 Driving licence

Jobs

2020- Óbuda University, Ybl Miklós Faculty of Built Environment
2014-2020 Szent István University Institute of Architecture, informatics department
2001-2014 SZIU-YMF Faculty of Descriptive Geometry and Computer Science, assistant lecturer
1999-2000 BMKE Department of Photogrammetry and Geoinformatics
2000- Geotronic Bt., executive (IT consulting, real estate management)

Educational experience

- Technical Informatics I/II (SZIU/YMK, 2001-2020)
- Computer representation (SZIU/YMK, 2001-2020)
- System administrator education (Szamalk, 2002)
- Computer networks (Soter Line 2004-2005)
- Informatics (Ybl, 2000)
- IT Basics (BUT, 2000-2001)
- Applied Informatics (BUT, 2000-2001)

Technical Experience

- Software development:
  - neural networks, deep learning technologies;

  - database programming;
  - solid understanding of design patterns;
  - functional programming, algorithms;
  - web-technologies (JS, XHTML, CSS, PHP, CGI programming)
  - expertise in Python, Bash, JS/ECMAScript, SQL
  - usable knowledge in haskell, C, C++, x86 assembly, ruby
- Computer system design and system administration:
  - 20+ years in Linux world as system administrator and power user;
  - open source technologies;
  - multiple years of experience in spam filtering and email systems;
  - experience of Linux operation in enterprise environment;
  - TCP/IP networks, routing, firewalls;
  - Open Source containers and virtualization technologies;
- CAD systems (AutoCAD, ArchiCAD)
- GIS systems
- professional translation experience (Linux Journal magazine)
- basic knowledge of hobby electronics and circuits, Arduino programming