DE GRUYTER
OPEN

10.1515/jbe-2015-0003

# Auditing Community Software Development

**Gergely Mészáros**

Institute of Civil Engineering, Szent István University, Budapest, Hungary
meszaros.gergely@ybl.szie.hu

***Abstract:*** *In accordance with European efforts related to Critical Information Infrastructure Protection, in Hungary a special department called LRL-IBEK has been formed which is designated under the Disaster Management. While specific security issues of commercial applications are well understood and regulated by widely applied standards, increasing share of information systems are developed partly or entirely in a different way, by the community. In this paper different issues of the open development style will be discussed regarding the high requirements of Critical Information Infrastructures, and possible countermeasures will be suggested for the identified problems.*

***Keywords:*** *Open Source, Critical Information Infrastructure, software security*

## 1. INTRODUCTION

The European Programme for Critical Infrastructure Protection (EPCIP) sets the overall framework for activities aimed at improving the protection of critical infrastructure in Europe across all EU States. In accordance with subsidiarity principles of EPCIP, the member states should establish their own frameworks of critical infrastructure protection [1].

As the necessity of organised assurance of Critical Infrastructure Protection (CIP) and Critical Information Infrastructure Protection (CIIP) became evident, new tasks were assigned to a public office or special purpose department. In Hungary the Act 2013/L. and the supporting Government decree No. 233/2013 (VI.30.) are establishing the conditions and role of Disaster Management in Critical Information Infrastructure Protection. According to this legislation dedicated department named "LRL IBEK"[1] has been founded under the National Directorate General for Disaster Management under the Ministry of the Interior (NDGDM). The LRL IBEK is integrated part of Hungarian CERT system.

The actively developing organization will, among other tasks, warn the operators of CII in case of intrusion from the global cyber space and continuously monitor for possible vulnerabilities related to security of a system in accordance with its scope of authority [2, 3].

In some important application fields like cloud computing or RDBMS, open-source softwares and libraries have matured significantly, and now can be seriously considered to replace commercial counterparts [4]. In case of CIIP, key selling points of open-source can be vendor independence and auditability. The most important, and far most popular type of open-source are collectively developed free products, often regarded as OSS/FS, FLOSS or FOSS. FLOSS is characterized by a distinct license, distribution method and development style all of which may have important security consequences [5].

[1] „Létfontosságú Rendszerek és Létesítmények Informatikai Biztonsági Eseménykezelő Központ"

This paper is not intended to be a pro-FLOSS proposal although one can argue many trivial benefits of closed source solutions like better support, unified environment, usability, total cost of ownership, compatibility and transition problems are not discussed here in details. This is intentional. The focus of this paper is on inherent virtues and major problems of FLOSS usage in CII with strong emphasis on security considerations.

While most software in CII are still proprietary systems, the emerging trend of open source usage frames the question: Are the current policies, monitoring and security audit methods well suited for a community developed open source systems? In my opinion this is not the case. Effective Software Security Audits -- often regarded simply “audits” in this article -- are crucial in high security environments. The audit process and supplementary concepts required for successful audits like proper security policy, penetration testing methodology and vulnerability assessment should involve as many information as possible. Disregarding special threats and opportunities regarding FLOSS there means certain information loss. This may be a problem because contrary the popular belief, open source is everywhere.

The goal of this paper is to summarize the current state of Critical Information Infrastructure Protection in Hungary, show examples of security issues associated with open source development model, and demonstrate possible solutions.

## 2. INFORMATION INFRASTRUCTURE PROTECTION

In 2009, “European Commission adopted a Communication on Critical Information Infrastructure protection (CIIP) focusing on the protection of Europe from cyber disruptions by enhancing security and resilience” [6]. Two years later, in March 2011, the Commission took stock of the results achieved that far and announced follow-up actions in the Communication on CIIP, followed by The European Parliament Resolution in 2012 on “Critical Information Infrastructure Protection: towards global cyber-security”. The cybersecurity strategy and proposal for a Directive on network and information security published in 2013. The four pillars of this model are: prevention and early warning; detection; reaction; and crisis management.

Critical Information Infrastructure (CII) not only forms one of the constituent sectors of the overall Critical Infrastructure, but also is unique in providing an element of interconnection between sectors as well as often also intra-sectoral control mechanisms. One problem concerning the protection of CII is that, it cannot be modeled entirely with usual approaches like reliability theory and fault tolerance. Deliberate attack of internal or external agent is also a factor. Furthermore, interconnections between CI elements lead to larger-scale and often unanticipated failures, particularly where interdependencies imply that infrastructures are dependent on each other and can propagate the failures from one sector to another [7]. In such a dependent and strongly interconnected environment protection of communication lines plays a crucial role.

Fortunately numerous countermeasures are developed in answering the emerging threat, some of which will be shortly summarized bellow.

### 2.1 Existing solutions and methods

According to the OECD recommendation, member countries should:

- Developing a national strategy that gains commitment from all those concerned, including the highest levels of government and the private sector.
- Taking into consideration interdependencies.
- Conducting a risk assessment based on the analysis of vulnerabilities and the threats to the CII, in order to protect economies and societies against the impacts of highest national concern.
- Developing, on the basis of the assessment, and periodically reviewing a national risk management process.
- Developing an incident response capability, such as a computer security incident response team (CERT/CSIRTs), in charge of monitoring, warning, alerting and carrying out recovery measures for CII; and mechanisms to foster closer cooperation and communications among those involved in incident response [8].

In Hungary, conforming with the EU efforts, most part of the recommendation is already fulfilled with Act 2013/L. coming into operation. Also, form July 2013, GovCERT-Hungary is the governmental CSIRT of Hungary. Designated by a Government Decree, this is national point of contact for international CSIRT and CIIP organizations [9].

The aforementioned LRL-IBEK will be operating in close cooperation with Hungarian GovCERT. The organization should:

- provide technical support and protection,
- help prevention,
- collaborate in information sharing and
- fulfill an educational role.

The department publish periodical security reports to national critical information infrastructures about the identified and published vulnerabilities.

As we can see, the organizational requirements are already forming or established, the main question remains if the existing system can or cannot be effectively used in the special case of FLOSS usage. The international CERT/CSIRT network usually doesn't differentiate between FLOSS and closed source (CSS), the vulnerability reports or response consider a specific vulnerable system or malware, disregarding the licensing or development style.

Considering the general CI organizational and operational security requirements, there are several well-established enterprise audit frameworks and standards available like Common Criteria, COBIT, ITIL, ISO/IEC 27000, which cover wide range of security requirements from policy to operation level. While pursue these standards may not be legal requirement, conforming to one or more of them is highly encouraged and widely used in critical environments. Unfortunately, the special development model of FLOSS often prevents expensive certification processes like Common Criteria, which means that the certification must be performed in-house. In fact, the current methodology of security audits are tailored to profit oriented corporations and align poorly with community development.

Without common language the security information sharing would be difficult. Fortunately, increasing number of standardization efforts trying to deal with the problem. The known vulnerabilities and weaknesses are standardized under a common identifiers like MITRE's CVE and CWE[2]. Emerging standards like STIX[3] and TAXII[4] can raise the cyber threat information sharing to the whole new level. STIX is a language used to communicate a set of cyber threat intelligence idioms, including threat actors, techniques, exploit targets, cyber observables incidents and courses of actions. [10]

### 2.2 FLOSS in disguise

One can argue that, few or no examples of FLOSS can be seen in CII environments, this way trying to handle its security issues is fruitless. In the following section we can see this is not the case. Today's complex information systems are using great number of open source components even if the organisation doesn't use any FLOSS applications officially.

In traditional closed source development style only limited number of developers can access the source code and the system's internal documentation. The exact methods and algorithms are often unknown. By contrast, products of community development are always transparent, from the source code to the issue tracking system. Often developers' internal communication is even available. It would be naive to suppose that all of this excellent source of algorithms are completely ignored by closed source developers.

Viewing FLOSS in a wider sense, the shared information and the unique development style may have impact on diverse set of applications in our system, including:

1) open source products,
2) publicly available source and configuration snippets and systems using them,
3) internal developments using FLOSS libraries,
4) closed source applications using FLOSS libraries.

While avoiding the first point may be simple question of decision-making, avoiding the others could be challenging or virtually impossible. Configuration snippet and source example usage can be regulated only by strict policy. The developer team also may use a special guideline not to use any FLOSS related product, however this decision would result in increasing difficulties by excluding large number of excellent libraries and frameworks.

Contrary, the built-in FLOSS code can not be dodged. Open source use is widespread in both private and government systems, and has been for many years. According to Mitre "Microsoft is one of many examples of commercial companies that make extensive use of open source software to build and expand their product line. Internet Explorer is an example of a notable Microsoft utility that is based heavily on OSS. [...] Google is another industry leader that uses OSS heavily both internally and in its commercial products." [11]

Verifying commercial applications in respect of FLOSS usage would be very hard. The organization must trust in external -- possibly foreign -- audit and auditors, which might not be appropriate in critical environment. On the other hand, vendors are using increasing number of open source, the FLOSS component usage is proliferating and soon become impossible to avoid [12, 13]. At the same time, the traditional security audit methods may be suboptimal in dealing with FLOSS.

Publicly available source repositories and issue tracking databases may be exploited to find unpatched vulnerabilities [14, 15].

[2] MITRE Common Weakness Enumeration (CWE) and Common Vulnerabilities and Exposures (CVE)

[3] Structured Threat Information eXpression.

[4] Trusted Automated eXchange of Indicator Information, transport protocol for STIX.

The vulnerable OS code may appear either in the organization's custom solution or any installed commercial application and the vulnerability can propagate to the CII. Without a proper dependency database it is very hard to identify the problem source. Auditing specific forms of community development like github-style pull-based model can be especially challenging since multiple clones should be analysed to find patched vulnerabilities which is not yet pulled into the mainstream version.

Our organisation's information system may be built entirely on proprietary solutions, and still we are using huge number of open source. Some CIOs[^Chief Information Officer] may say this is not our problem, we have SLA[^Service-level agreement] with our suppliers we just have to enforce obligations. While this may be valid argument in enterprise environment, it is unaccaptable in case of CII, where terrorist activity or nation state operations may inflict far more problems than mere financial damage.

### *2.3 Projects targeting FLOSS*

Several attempts were made to deal with FLOSS's special security and maturity issues. A notable example is the FLOSSMetrics research project, funded by the European Commission, which construct, publish and analyse a large scale database with information and metrics about FLOSS development indexing several thousands of software projects [16]. The U.S. Department of Defense also spent years creating three documents analyzing and elaborating the role of OSS in DoD systems [11].

These projects targeting open source maturity and usage feasibility, clearly identify FLOSS as an important and valuable resource. However currently the velocity of development seems to be slowed down.

There are also commercial firms providing specialized FLOSS audits. For example Black Duck Open Hub (formerly Ohloh) provides extensive searchable database of open source projects. Security scans across application portfolios to find and remediate open source vulnerabilities is also available.

## 3. MITIGATION POSSIBILITIES

In order to secure CII containing FLOSS applications or components, special countermeasures can be established. As we saw previously FLOSS component usage can not be entirely avoided, almost all modern systems contain more or less open-source code.
In this section some possible mitigation strategies will be summarized.

### *3.1 Dependency tree analysis*

Current open source operating systems are using integrated software repositories employing some sort of dependency checking ability. Currently Windows 10 also includes a Linux-style package manager named OneGet. The package manager can be used to identify the actual versions of software and its specific dependencies like libraries. Unfortunately this information is only available in case of purely open source projects where the source code and the compile options used would clearly identify the included library versions. However some dependency information can be extracted directly from the compiled binary either via header analysis or sophisticated pattern matching while bytecode-based and Just-in-time compiled languages like java can be systematically analysed in their compiled form. Free tools like Dependency Finder[5] are available that can extract dependency graphs and mine them for useful information.

Most FLOSS licences require the original license file to be included in the distributed product. Conforming vendors must provide these licenses, this way many included libraries can be identified. Unfortunately this legal requirement is often violated [17] what can be a major obstacle in extensive analysis.

All collected dependency information should be aggregated into a global dependency database. When a new vulnerability is reported or identified by repository scanning, a database search can quickly reveal all affected systems. This could be highly advantageous since short response time is critical in case of a newly released vulnerabilities.

In order to build most exhaustive database possible, suppliers of CII should provide all necessary library dependency information regarding their product. Optimally this requirement should be enforced by a contract.

### 3.2 Open Source Policy

The organization should create a clear security policy regarding FLOSS usage. The policy should cover alike the organisation's supply chain, internal development and usage. Available open source related policies should be collected from external developers and CSS vendors. Without it the organisation is unable to assess its level of contamination and protection mechanisms in place regarding FLOSS usage. It should be clear whether open source components are allowed in the development, support systems and end-user side. Practices of openly available code and configuration snippets should also be controlled.

There should be an emergency plan to handle serious security event affecting FLOSS components used or may be used in the organisation systems. As we saw in the previous section, the involved systems may well be proprietary ones. The policy should be clear regarding what can be done in such situation. Is it possible to change the source code and recompile the components, or should we consult the vendor's open source policy or I have to shut down the application immediately? How can we identify our affected systems? Questions like these should be quickly answered in emergency situations.

To be able to check the vendor's policies is important because we can understand whether they are capable to handle such situations or at least able to identify the problem or not.

With well thought open source security policy in place, the threats posed by FLOSS components can be greatly mitigated.

### 3.3 Central FLOSS repository

Many Open Source product distributed in binary form, using a package manager infrastructure or in some form of standalone precompiled package.

[5] http://depfind.sourceforge.net/

Using precompiled binaries constitute unwanted dependency on the distributor. From security perspective it is better to download the source code from trusted sites and compile it to the organization specific needs [18].

However validating and compiling the source is a tedious process requiring time and special knowledge, which may be beyond the resources of the CI operator. In order to resolve the problem, central FLOSS repository governed by a trusted party like LRL IBEK can be used for new installations. Distribution of signed binary packages is proven to be working solution in case of multiple Open Source distributions and can be easily set up.

Assuming that the Operator is obligated to use the central repository with a sufficient update-policy in place, several security issues can be avoided including improper compile option usage, code change by malicious third party and vulnerable version usage.

While keeping large number of applications up to date is overwhelming, central storage of limited set of popular software like network management tools, web-servers, CMS and e-learning systems may be well worth the added complexity.

### 3.4 Active government participation

Recent critical vulnerabilities in popular open source libraries draw attention to the importance of the risks of inadequate support. For example a vulnerability in OpenSSL library known as 'Heartbleed' opened two-thirds of the Web to eavesdropping for two years before patched. The project was found severely underfunded despite its widespread usage [19, 20].

FLOSS projects can be seen as public value, therefore eligible for support in its own right. Furthermore, the open source philosophy conforms with the open government principles, participation and transparency [21, 22].

Support may be a form of legal or financial allowance or direct contribution like comments, security patches, feature developments or audits. The in house security patches and audit results should be shared with the community if possible. A large scale example of a government participation open source development is the SELinux kernel security module, which was originally contributed by the United States National Security Agency (NSA).

Active government participation results in added trust and increased stability, from which all participants can profit.

## 4. SUMMARY

The increasing acceptance of open source libraries has significant impact on application security considerations of recent information systems including CII. Current audit methods primarily developed for cathedral style commercial software and systems and may not be well suited for FLOSS audits.

Organizations concerning CII security like the recently formed LRL-IBEK, may consider extending their activities in order to achieve better insight of open source related security issues. In this paper three concepts are suggested which can further tighten the relations between the community and security audit of FLOSS components. Open source library dependency analysis may reveal uncovered vulnerabilities which can be especially effective in combination with forced disclosure of open source assets in products from external vendors. Promoting a central repository of common set of audited FLOSS versions can improve overall security, and may help avoid several issues related to multiple versions of the same product. Lastly, I would like to emphasize the importance of governmental participation in FLOSS development. Without that, the independent and secure information infrastructure required by CI can be hardly established.

## REFERENCES

[1] *Critical infrastructure,* 2015. <http://ec.europa.eu/dgs/home-affairs/what-we-do/policies/crisis-and-terrorism/critical infrastructure/index_en.htm>

[2] **Bognár Balázs:** *A létfontosságú rendszerelemek azonosításának, kijelölésének folyamata, az LRL IBEK működésének eddigi eredményei, a BM OKF elvárásai az NKE képzésével kapcsolatban,* [2015-05-12] <http://vtki.uni-nke.hu downloads/tk/IBOT_PILOT/PLENARIS/Dr_Bognar_Balazs.pdf>

[3] **Haig Zsolt:** *Információ – társadalom – biztonság,* Budapest: s.n., 2015. ISBN 978-615-5527-08-1.

[4] Donald Feinberg, Merv Adrian: *The State of Open-Source RDBMSs,* 2015. <http://www.gartner.com/technology reprints.do?id=1-2DTR05J&ct=150421&st=sb>

[5] **Mészáros Gergely:** Security impacts of community based software development. In: *CEE e|Dem and e|Gov Days 2015*. 2015: 2015. ISBN 978-2-85403-308-0.

[6] *Policy on Critical Information Infrastructure Protection (CIIP),* 2013. [2015-06-02] <http://ec.europa.eu/digital agenda/en/news/policy-critical-information-infrastructure-protection-ciip>

[7] **Javier Lopez, Roberto Setola, Stephen D. Wolthusen:** Overview of critical information infrastructure protection. In *Critical Infrastructure Protection*. Springer, 2012. pp. 1–14. ISBN 978-3-642-28919-4.

[8] ICCP Committee: *OECD Recommendation of the Council on the Protection of Critical Information Infrastructures*, 2008. OECD Council. <http://www.oecd.org/sti/40825404.pdf>

[9] *GovCERT-Hungary* . [2015-06-08] <http://www.cert-hungary.hu/node/1>

[10] **Sean Barnum:** *Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression (STIX)*. 2013.

[11] The MITRE Corporation: *Open Source Software* . 2013. [2015-06-04] <http://www.mitre.org/publications/systems engineering-guide/enterprise-engineering/engineering-informationintensive-enterprises/open-source-software>

[12] **Phil Marshall:** *OSS GovernmentManagement using COBIT 5* . 2012. [2015-05-23] <https://www.isaca.org/Education Online-Learning/Documents/OSS-Government-Management-using-COBIT-5.pdf>

[13] **Jai Vijayan:** *Growing Open Source Use Heightens Enterprise Security Risks - Dark Reading* . 2015jan . [2015-06-03] <http://www.darkreading.com/growing-open-source-use-heightens-enterprise-security-risks-/d/d-id/1318767>

[14] **J. Jang, A. Agrawal, D. Brumley:** ReDeBug: Finding Unpatched Code Clones in Entire OS Distributions. . IEEE, 2012. pp. 48–62. [2015-05-06] ISBN 978-1-4673-1244-8, 978-0-7695-4681-0. <http://ieeexplore.ieee.org/lpdocs epic03/wrapper.htm?arnumber=6234404>

[15] **D. Brumley, P. Poosankam, D. Song, J. Zheng:** Automatic patch-based exploit generation is possible: Techniques and implications. In: *Security and Privacy, 2008. SP 2008. IEEE Symposium on* . IEEE, 2008. pp. 143–157. [2015-05-06] <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4531150>Hogyan lehet automatikusan meghatározni a sérülékenységet a patch elemzésével.

[16] **I. Herraiz, D. Izquierdo-Cortazar, F. Rivas-Hernández, J. Gonzalez-Barahona, G. Robles, S. Duenas-Dominguez, C. Garcia-Campos, J.F. Gato, L. Tovar:** Flossmetrics: Free/libre/open source software metrics. In *Software Maintenance and Reengineering, 2009. CSMR'09. 13th EuropeanConference on* . EEE, 2009. pp. 281–284. [2015-06-03] <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4812771>

[17] **A. Mathur, H. Choudhary, P. Vashist, W. Thies, S. Thilagam:** An Empirical Study of License Violations in Open Source Projects. In: *Software Engineering Workshop (SEW), 2012 35th Annual IEEE* . IEEE, 2012. pp. 168–176. [2015-06-08] <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6479814>

[18] SANS Institute: *Security Concerns in Using Open Source Software for Enterprise Requirements*. 2003.

[19] **Robert Mcmillan:** *How Heartbleed Broke the Internet — And Why It Can Happen Again* . [2014-04-14] <http://www wired.com/2014/04/heartbleedslesson/>

[20] **Jack Wallen:** *From underfunded to funded within a heartbleed* . 2014. [2015-05-20] <http://www.techrepublic.com article/from-underfunded-to-funded-within-a-heartbleed/>

[21] **T.M. Harrison, S. Guerrero, G.B. Burke, M. Cook, A. Cresswell, N. Helbig, J. Hrdinová, T. Pardo:** Open government and e-government: Democratic challenges from a public value perspective. In: *Proceedings of the 12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times* . ACM, 2011. pp. 245–253. [2015-06-08] <http://dl.acm.org/citation.cfm?id=2037597>

[22] **Alexis O'Connor, Kian Win Ong, Ted Sander, Matt Ferlo:** *Government Policies on Open Source* . 2010. <http://www cs.washington.edu/education/courses/csep590/04au/clearedprojects/Ferlo.pdf>