**Auteur:** Ellen Wesselingh is senior architect at Fox Crypto, with subject matter expertise in product security certification, Member Workfield Committee HBO-ICT bachelor NSE program in Delft and Member Workfield Committee HBO-ICT bachelor Computer Science program in Rotterdam. The full article can be found on https://foxdatadiode.com/news/

# Eliminating the weakest link

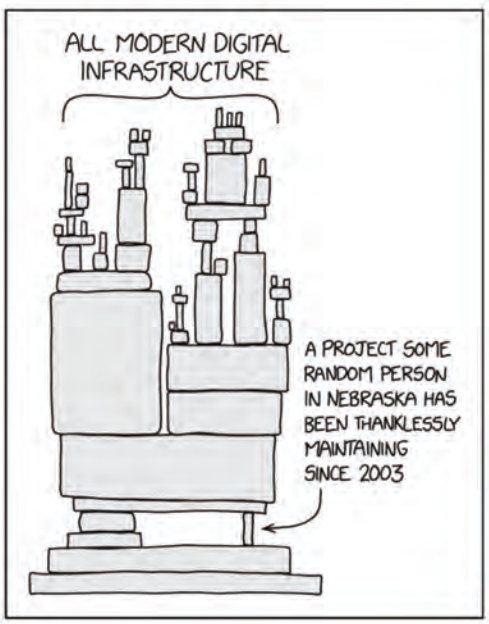
## Managing Supply Chain Cybersecurity Risk through Components Analysis

This article describes a follow-up activity for analysis described in Eliminating the weakest link – Managing Supply Chain Cybersecurity Risk through Life Cycle Modelling (IB-Magazine 4 2024): the analysis of security critical components, and follow-up action. Not all components in a system are security critical, so measures in the supply chain may vary in rigour. An example are the screws with which a casing is closed, provided the casing is not essential for security measures, such as anti-tamper. Hence, the security critical functions, being those functions that are security enforcing and those being security supporting, must be identified. After the supply chains for a product have been identified, measures may be discussed with partners. Open source projects may have to be treated differently.

In the previous article (IB-Magazine 4 2024), a supply chain risk analysis model was introduced, based on a MITRE model dating from 2013. Where the primary focus was to develop a supply chain attack framework that addresses the primary process in which a product with IT components is produced, this article builds on that model to further refine the risk analysis.

When developing a system, whether it is an embedded system with hardware, firmware, and software components, or a software only system, many supply chains exist. Randall Munroe (XKCD) showed this very profoundly for Open Source Software (OSS), in a 2020 cartoon (1). The picture illustrates the problem with many supply chains. In commercial as well as open source supply chains, the entity that finally brings the product on the market has little overview on, and control over, the many suppliers.

Some suppliers are large corporations with the means to implement cybersecurity measures in all their development processes. Other suppliers are on the opposite side of the spectrum: small communities without the means or the




Dependency (Munroe), source (1)

# Supply chain attacks are a real risk, and come in new formats frequently

knowledge to implement and maintain cybersecurity measures. Many more are probably in between those ends of the scale. The challenge is, that the entity bringing the product on the market, probably does not know exactly the status of each supplier.

Analysing all components of a product, and their respective supply chains, is probably impossible. Therefore, the scope of the components under analysis must be reduced. This is done by eliminating non-security critical components from the analysis. How this can be done is the topic of this article. Secondary services such as the supply chain of other office or communication processes are not in scope, but may be addressed in future work. The same holds for supply chains of services that are provided to customers.

### Methodology used

The original research was inspired by a presentation from Andrew Huang for a Blue Hat conference. In his presentation, Huang mentioned a number of attacks. The articles describing those attacks were analysed for further ideas and references (snowball method). Subsequently, a search for (hardware) supply chain attacks with more generic keywords was conducted to find other supply chain hardware attacks. These attacks were then analysed. No specific search for software issues was done in this phase because of the overwhelming amount of software security problems.

Some seminal software attacks were studied for illustration of supply chain software attacks. In 2024, a next level type of open source attack emerged (XZ Utils): a combination of social engineering and poisoning of open source libraries with backdoors. The attack did not materialise in full, but was very close to success. Other examples of supply chain problems that emerged in 2024 are the worldwide failure of Windows servers due a CrowdStrike update in July, the temporary shutdown of the Dutch high-security network NAFIN in August, and the pager/walkie-talkie attacks on Hezbollah in September. The Windows and NAFIN problems were probably not the result of an act with malicious intent, the resulting system failure caused widespread problems for society nonetheless. Finally, the attack on Hezbollah was an elaborate supply chain attack, in which an Advanced Persistent Threat (APT) was capable of manipulating a complete supply chain.

The XZ Utils backdoor showed that a malicious entity was capable to create a persona (or multiple), and build trust with that random person thanklessly maintaining the library since 2003. The malicious entity was eventually able to take over the project, and implant a backdoor to target security critical software like OpenSSH. In this case, an application built on top of the core was able to infect another component through that core. One of the largest cyber incidents in history was avoided because one developer running Debian Sid Linux noticed the SSH Daemon using a bit more CPU time than normal, while benchmarking something completely unrelated. The type of attack shows the importance of knowing your supply chains. It also shows how difficult it is to manage all supply chains.

The model as described in the previous article is the starting point for the next step in the process: the determination of security critical components, and detailed analysis of the supply chains of these components. For this part of the research, the Common Criteria is used. This international standard for IT product security evaluation defines the notion of security enforcing, security supporting, and security non-interfering functions.

## Previous model

The model as described in the previous article uses the following step-by-step supply chain risk analysis:

1. Sketch a life cycle and if possible, a supply chain model.
2. Determine the applicable categories from the attack surface categories: architecture, standard, hardware, firmware, operating system or software, development environment.
3. For the above determined applicable categories, determine the attacks that are applicable.
4. Plot the attacks on the life cycle model.
5. For each attack, discuss with expert colleagues what countermeasures are possible to counter the attack.
6. For all countermeasures, discuss with the relevant supply chain partners what is possible.
7. When all countermeasures that will be applied are known, identify any remaining risk.
8. Within a specific evaluation / certification framework, the risk may be quantified. For example, in the Common Evaluation Methodology there is a vulnerability analysis, of which the outcome is the calculation of attack potential needed to attack the IT product under evaluation successfully.

The steps described above, are the generic steps for any supply chain analysis. However, there are many types of products with their own supply chain specific issues. Therefore, for a specific product, a more detailed analysis is necessary. This analysis requires to narrow the scope to keep the workload manageable. The first step is to define for each component whether it is:

1. Security enforcing: these components directly implement a security functional requirement, for example user identification and authentication.
2. Security supporting: these components do not directly implement a security functional requirement, but have to function correctly for the security enforcing components to be able to do their job. A typical example is the Human Machine Interface (HMI), which is used for user identification and authentication.
3. Security non-interfering: the security enforcing and supporting components do not rely on these components for their correct functioning. Ideally, these components cannot influence the correct functioning of security enforcing and supporting components at all. The XZ Utils case showed just how difficult this is. Examples are usually the casing and power connection.

## Next steps in the analysis

The definition of a component as security enforcing, supporting, or non-interfering, is dependent on multiple factors. The first, and main, input is the definition of assets in the security claim. This claim should describe what the type of assets are, such as (personal) data, or operational processes. For the defined assets, it must be clear what protection is to be provided: confidentiality, integrity, or availability. Typically, for (personal) data, confidentiality is the most important aspect. For operational processes, availability is usually the most important aspect. Both confidentiality and availability require system integrity as a fundament.

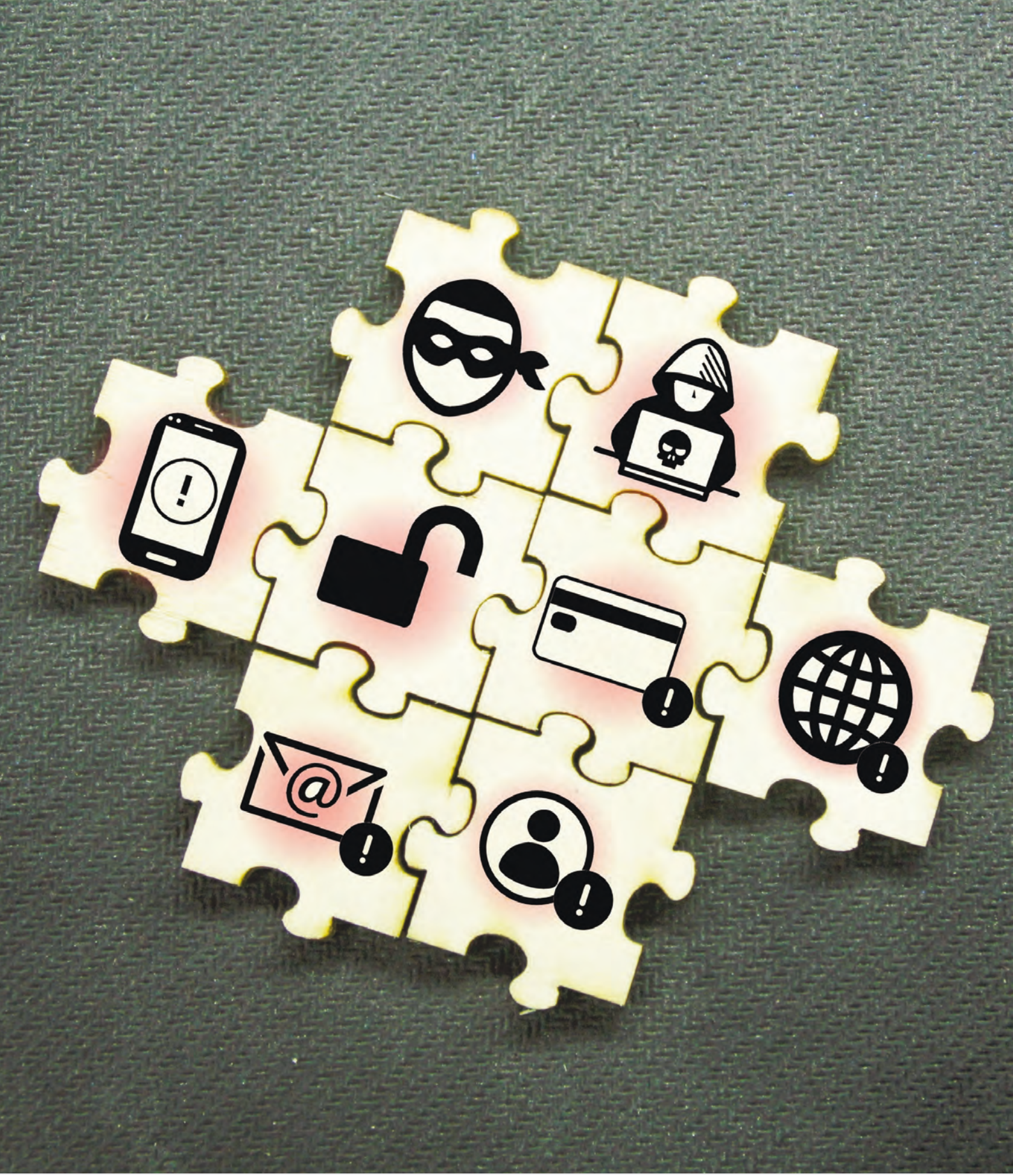
The second, and very important, input is the security functionality that the product has to provide. The fact that the security claim is used for analysis, means that certain quality criteria apply to that claim. If a security claim description is not factual or measurable, then it is not possible to base the analysis on that claim. Common Criteria provides a library with standardised security functional requirements to support the development of a well-defined security claim.

The experts decide for all components their contribution to the security functions. This is usually done in the format of a meeting with experts from different disciplines. For each component, its function with regard to the assets as defined in the security claim, is discussed. This leads to a list of components classified as either security critical (enforcing or supporting), or non-interfering. Security non-interfering components do not have to be subject to precautionary measures in the supply chain, security critical components do.

For these security critical components, the manufacturers are determined. A manufacturer can be a commercial entity, established in the EU itself or through a partner. This type of supplier can be addressed in a formal contractual relationship. A manufacturer can also be an open source project. With this type of supplier, no formal relationship exists. The community maintaining the open source project may even be unaware of the fact that specific companies use their products. Therefore, this type of supply chain needs to be treated differently.

## Managing Supply chain partners

When dealing with formally established entities, supply chain security measures can be negotiated and defined in the contract. Agreed security measures, comparable to a Non-Disclosure Agreement (NDA), would help to formalise the supply



chain relationship between supplier and acquirer. As an input to this contractual agreement, the measures as described in the appendix of the previous article can be used.

When dealing with an open source community, establishing a formal relationship may prove to be hard. Fox-IT has experience with this process, when building OpenVPN-NL on top of OpenVPN. OpenVPN exists since 2001, and is maintained by an international community. OpenVPN-NL exists since 2011, and is maintained by Fox-IT. It took several years to establish a relationship of trust between both parties. There are other examples of commercial companies supporting open source projects, for example by giving employees with specific expertise time to maintain an open source project. Other communities are strong communities themselves, built in the course of years.

Sponsors could play a role in building stronger OSS communities. Commercial entities may be willing to participate in the development of open source projects under the umbrella of Corporate Social Responsibility (CSR), but maintaining a large open source project is probably more expensive than the CSR budget. However, when multiple companies donate, the burden is shared. In 2011, Fox-IT published OpenVPN-NL open source, Fox Crypto is maintaining it. In 2022, Fox-IT made its own toolset for incident response analysis, Dissect, open source. In 2023, Fox Crypto built, and published open source, its' own implementation of a Post Quantum Resilient XMSS algorithm. Parts of these project have been sponsored by the Dutch government. This sponsorship enables the Fox companies, which are medium businesses, to help in building a more (cyber)secure society.

How, as a company, to build trust in an open source community? In our experience, communication and participation are crucial. Trust must be mutual. Companies often support with technology (code), but can also help a project to make their development processes more mature. For example, by contributing to secure development processes or providing infrastructure or tooling to perform code quality checks. Building trust typically requires showing capabilities and long-term commitment. Over time, a community will be able to judge company contributions on their merits, and the company can judge whether the stability and maturity of the community meets their supply chain requirements.

Basically, everything that the attacker in the XZ Utils case did… In hindsight, it is somewhat strange to see that a company that builds trusted systems for the high assurance industry is met with suspicion, while entities claiming to be an individual are not rigorously checked at all. Given the history of government meddling in cryptographic issues, it is understandable. But companies are usually not lead by malicious intent with regard to building backdoors on purpose, the problems probably arise due to incompetence rather than maliciousness.

### Steps to assure if the components fit

What should manufacturers of composite systems do, when incorporating open source components in their product? They should at least assess whether these components are security critical. For those components, the following checks may provide some assurance that the components are fit for purpose:

1. Whether the community is a robust community, in the sense that it is well established, and maintained by a sufficiently large pool of maintainers. With regard to maintenance, active maintenance can be monitored using the following metrics: the time it takes to solve reported issues, how many pull requests are currently open, average time a pull request remains open.
2. If possible, verify where the community is established, and where the maintainers reside. Find out which organisation maintains the community, are one or more companies or other established organisation involved, or is the community maintained purely by volunteers. Do the organisations supporting the community have a solid reputation. Team size is also an indicator of the stability of the community. Refrain from using open source software from communities established in nations with an offensive cyber-strategy.
3. Review the quality of the product, the accompanying documentation, and the work processes. Check whether the product is feature complete, if not, what the status of the product is. Check whether the documentation is available, readable, and complete. Verify if issues are reported and fixed on for example GitHub, and if the time between reporting of an issue and solving the issue is commensurate with the functionality of the product and the severity status of the issue. Verify if the product is actively discussed on platforms like Discord, Reddit, Stack Overflow. If there is a product roadmap available, this provides assurance about long term availability.
4. If possible, use externally verified components. Some OSS have been evaluated, which provides some assurance about the quality. The OpenVPN-NL variant of OpenVPN is

such an example.

5. When the technical expertise is available, do a source code review. As source code reviews are laborious, focus on the security critical components, such as input validation or cryptographic operations. Another option is to verify that public (high quality) code reviews have been done. OpenVPN for example is audited externally every now and then.
6. Finally, check whether the OSS license is fit for purpose. There are license types that prohibit certain types of use of the product, or that require open source publication of any changes to the product.
7. Once an OSS component has been chosen, incorporate open source vulnerability scanning in the work process for importing it. This does not stop zero day attacks such as the XZ Utils attack, but it limits the potential attack surface.

These steps require resources. This may be viewed as a cost factor, where it should be viewed as an investment factor. It starts with the realisation that a security breach will certainly happen when security has not been properly reviewed during the development process. Security breaches tend to be very costly (see below).

## Final steps

When the security critical components have been identified, and the applicable countermeasures are agreed upon with the supply chain partners, or otherwise addressed, remaining risks can be identified. For this step, a suitable risk identification method for the specific industry should be used. The Common Evaluation Methodology (CEM) provides a generic method for products, by modelling the attack potential that is needed to successfully attack a product. For Integrated Circuits (IC) and similar devices, there is a specific methodology available. However, these methods are laborious. For low risk Information Technology (IT) products, a method using Security Integrity Levels along the axes of exposure and impact may be sufficient. For Operational Technology (OT), a method using exposure, harm, in combination with controllability, should be used. For example, automotive safety standard ISO26262 uses Automotive Safety Integrity Level (ASIL). This model might be transposed to cybersecurity.

The remaining risk then has to be assessed for risk mitigation with the implementation of additional measures, or risk acceptance. The level of risk should be assessed against the cost of further risk reduction. Weighing risk against (further) mitigation is ultimately a business decision. This requires understanding of cybersecurity risks by the business. Seminal cases, such as Diginotar, University of Maastricht, and Maersk, which have been publicly discussed, may help the business in assessing the potential cost of a cybersecurity breach. In the case of Maersk, the entire global internal network had to be rebuilt. Not only does this lead to the cost of having to rebuild the network, it also leads to serious cost due to lost productivity. The latter may have been the largest factor in the total cost in this case.

## Discussion and Conclusions

Supply chain attacks are a real risk, and come in new formats frequently. Analysis of the attack surface, to be reduced by countermeasures, is the starting point for mitigation. This article discussed further steps in the analysis of components provided by partners in the supply chain. While not all supply chain risks may be mitigated, thinking about your supply chains and options for risk mitigation are an important step towards a more robust supply chain from a security perspective.

The use of well-established standards and methods helps in establishing a base line with regard to cybersecurity in the supply chain. For small companies, or companies with little cybersecurity experience, this will be a challenge. These companies may start by using cybersecurity certified products, when available. These products are often more expensive than non-certified products, because certification requires effort and hence, a business case. However, a breach in the security of a product can prove to be very expensive. A famous example of just how expensive a breach can be, is the Dutch Diginotar case. The company eventually ceased to exist, after it was hacked. The Maersk breach with NotPetya lead to an estimated cost of $300 Million, and this company was not the only victim that suffered severe damage.

The model and the next steps as described in this article, provide a basis for supply chain risk analysis and mitigation. Important steps to make are to identify the supply chains, to identify the security critical components suppliers in the supply chain, and to make formal arrangements with those partners. The exact implementation of the countermeasures is not part of the model, as these are at the discretion of the supply chain partners.

## Reference

(1) Randall Munroe (XKCD). Cartoon #2347 Dependency. August 17, 2020. https://xkcd.com/2347/ (CC-BY-NC 2.5) and https://www.explainxkcd.com/wiki/index.php/2347:_Dependency (visited 11 June 2024)