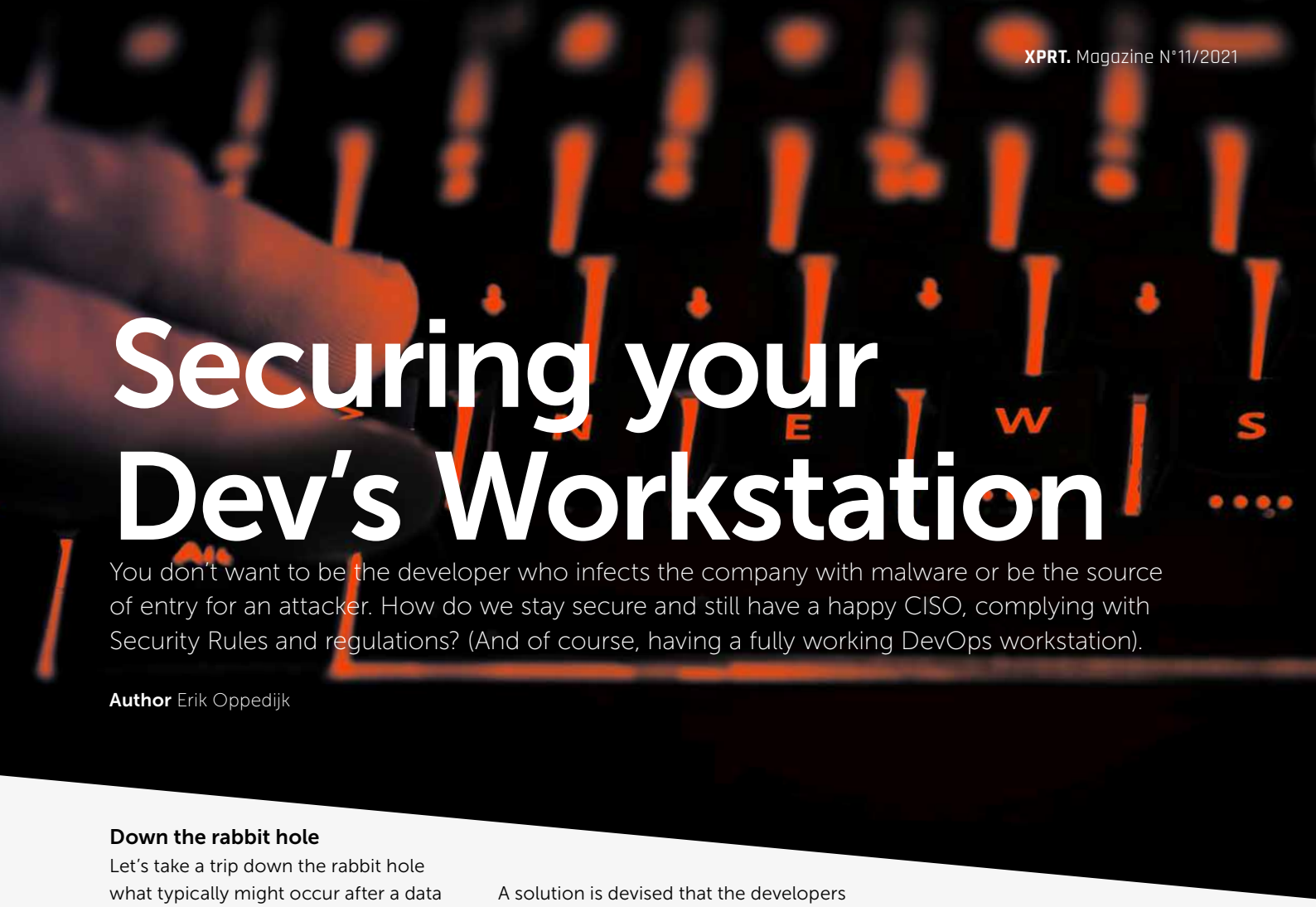


Securing your Dev's Workstation

You don't want to be the developer who infects the company with malware or be the source of entry for an attacker. How do we stay secure and still have a happy CISO, complying with Security Rules and regulations? (And of course, having a fully working DevOps workstation).

Author Erik Oppedijk

Down the rabbit hole

Let's take a trip down the rabbit hole what typically might occur after a data breach/security incident. Management or a CISO might ask you to remove the Local Admin permissions from your laptop.

Without Local Admin, installing and updating some software is harder, so we need to rely on a support team to (quickly) package new software versions for the developers. Of course, this slows down as the support team isn't able to keep up with all packaged application updates.

Then "they" find out that the developers still run plenty of portable apps* (which don't require admin privileges to run).

This leads to a complete "Application Allowlisting"** scenario, where only a handful of approved applications is allowed to run. Of course, this slows down the developer productivity since no compiled executable can be made to run and any tool used must first be allowed by the security team.

A solution is devised that the developers should work from a VM or docker container. This in turn can be used by the developer for all kinds of things, including day-to-day tasks like reading his e-mail or installation of non-work-related software.

Finally, "they" find out that the VM/docker container with full access is used by the developers for all kinds of software. So, it is back to square one, "they" require the removal of Local Admin permissions from this VM/docker... and we start again at the top.

Take a step back: Looking at risk management

If we take a look at the developer population, we can divide them into several groups:

- Developers/Contributors to Code (Low Privileged Accounts)
- Project/Pipeline administrators (Medium/High Privileged Accounts)
- Production Access (High Privileged Accounts)

We want to utilize Privileged Identity Management (PIM) for the Medium/High Privileged accounts and for production access, so every time a user needs these permissions, an elevation is required. This would also be the group of people working with the most sensitive secrets and intellectual property/trade secrets.

This leaves the developer group with access to the source code, which, in a typical organization, does not contain any military secrets or extremely confidential source code. Our DevOps pipeline and four-eyes principle on check-in/merge already provides us with a nice first defense line.

As we saw in "Down the rabbit hole", blocking (or also called Application Allowlisting) isn't working very well for all developers, so we need to take another approach: Detection.

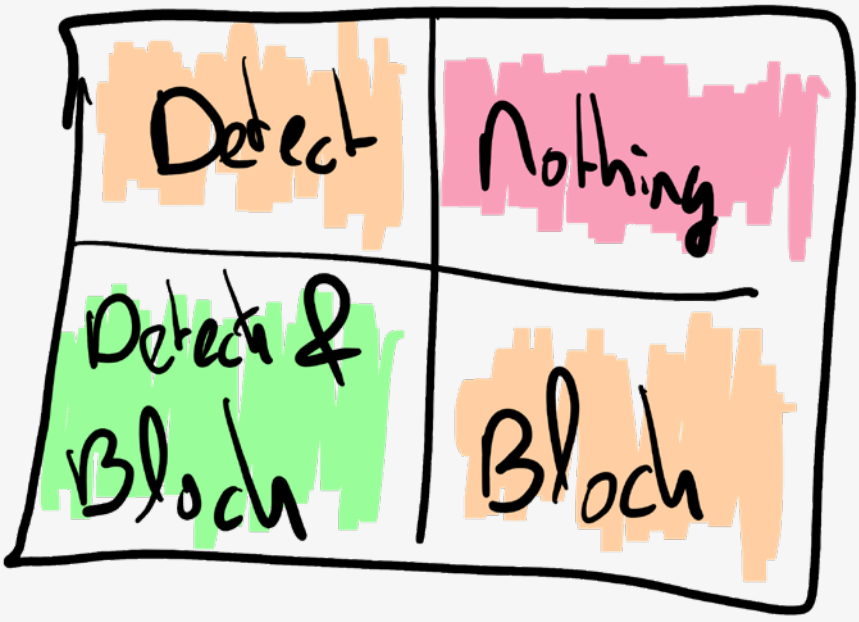
* Portable Apps are executables that allow to be run from read only or non admin locations, just from a user folder. Examples of this could be the user installation of Chrome, or tools like TeamViewer portable or 7-Zip portable.

** Application Allowlisting is only allowing certain applications to run, based on a hash value of the executable, any change (by an attacker or software update) will invalidate the hash and cause the application to be blocked.

With Detection, the developers can run all the things they want on their laptops, with an advanced "Endpoint Detection and Response (EDR)" tool available to detect malicious behavior. The EDR tool can spot suspicious processes/memory injections and detect connections to suspicious IP addresses. It works by looking at unusual behavior on the system.

With the EDR Tool in Detection mode, developer productivity is not harmed as it would be in the Block mode, which is used for regular users.

The biggest risk usually consists of unpatched software, which is often targeted by phishing attacks. Our best line of defense starts of course with educating the developers by informing them that they are the ones being attacked.



One of the ways is to patch all software on the developer's machine, coupled with the detection capability of an EDR, which should provide a nice line of defense.

Here come the auditors

The management/CISO go along with the proposal on how to secure your developer workstation, but quickly they start asking questions such as:

- Is the IT auditor pleased with this situation, how do the developers keep their machines up to date, and how does this fit in with the company policies which disallow everything, and more.
- How do we escape from this rabbit hole?
- If we take a look at the international ISO 27001 standard, there is a separate chapter (14) about software development in Annex A.[1]
- As with all standards, it is very important to read them and to understand the different terms including: **must**, **should**, **consider**, **depend**, **appropriately**, etc.

For instance, in A.14.2.6 Secure Development Environment, we should consider the sensitivity of the data, risk assessments, business/legal requirements. Back to our original assumption:

If the developer is not working with live production data, and is not working on extremely valuable code, then we don't need to take the same steps we take to protect our sensitive data/documents. According to the standard, we need to appropriately protect the environment, and not constrain it at all costs!

We need to classify code as "internal" and not as "top secret", because the secrets should not be accessible by all developers. With the concept of enterprise inner source (internal open source), almost all source code should not be sensitive. This allows you to focus on the real sensitive pieces, like the DevOps pipeline, or that single team that manages the code of your trade secrets.

Solutions

How do we solve our "problem"? Training and awareness should always be step number one, otherwise it is like rearranging the deck chairs on the Titanic. In addition, We need a combination of tooling and processes.

Tooling

There are several tooling options available to help remediate the problem:

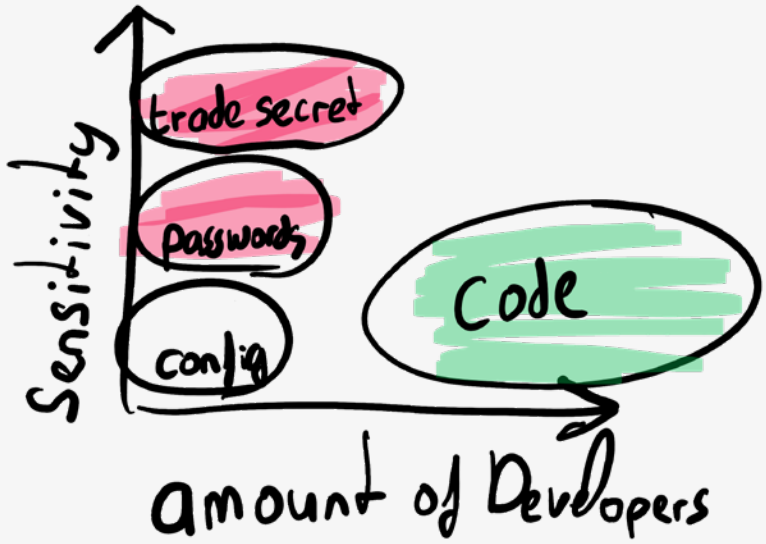
- Identity Protection (AAD P2 feature);
- Privileged Identity Management (AAD P2 feature);
- Defender for Endpoints (previously known as Defender ATP, not to be confused with Defender Antivirus which is a completely different product).

With Identity Protection we can specify conditional access rules based on risky behavior, e.g. a foreign logon location, change of browser, or a sudden location change during a session.

Privileged Identity Management is what we need when we need to elevate our permissions to perform a Medium or High Privileged Action, this is the Least Privileged concept: don't run as a High Privileged account by default.

The last one is Defender for Endpoints – this is an Endpoint Detection & Response (EDR) tool, which can detect suspicious behavior on the machine, like suspicious IP connections, running process modifications, but also vulnerable installed software.
EDR combines Alerting (and blocking/quarantine) together with Vulnerability Management capabilities to secure the endpoints.

[1] https://www.isms.online/iso-27001/annex-a-14-system-acquisition-development-and-maintenance/



Processes

The following processes are relevant:

- marking the developers as Priority investigation employees during SOC alerts.
- patching and automatically inform the developer on unpatched software and packages.
- The Security Operations Center with knowledge of development.

Whenever a security alert is processed by the Security Operations Center (SOC), the alerts for developers should receive investigation priority over regular users in the organization.
This ensures that suspicious behavior, like packages/scripts downloading extra content from the internet, is investigated with priority.

The best line of defense is keeping the machine up to date (See: background on patching). Tools like Defender for Endpoint can detect the vulnerable software, so that a workflow or preferably automation runs to directly inform the developer of the issues on his/her machine. This direct feedback loop is much better than having a monthly report being sent to the CISO on the state of all machines.

The last success factor is having a SOC team with development knowledge. How else can a series of suspicious activities followed by a flood of network connections be attributed to an attack, or just the test runner framework being used?

Best practices: background on patching and removing Local Admin

We all know that we need to patch our software, but only when we are not right in the middle of a refactoring session.

Combine this with running as a Local Admin and we have a potential disaster waiting to happen.

But what exactly is the impact of removing Local Admin, according to this research[2], of the 192 critical vulnerabilities on windows, 102 would be stopped by removing Local Admin permissions.

Applying system hardening (especially blocking process creation from Office or through WMI, very often used by malware) is another best practice for reducing the likelihood of spreading attacks. Hardening steps can be gathered from the Center for Internet Security (www.cisecurity.org) or if you've deployed that Endpoint Detection and Response (EDR) product, it will show you recommendations to beef up the security of your system.

But if we look at the total of critical vulnerabilities(192), only 2.5% is used in the wild to take over machines. This still leaves 5 critical items to fix, and they can be fixed by patching your machine. **There is no excuse for not patching your system.**

Quick patching is the best defense against almost all threats, so don't delay installing those patches for a long time.

Summary

Patching, patching, patching, just patch your machines, no excuses! Combine this with an alert system from the EDR where you as the developer directly receive the alert of out-of-date software and missing OS updates.

Don't run as admin by default on your DevOps workstation, run as a normal user, and make sure you can use that Local Admin account to temporarily elevate your permissions to Local Admin. (Just make sure that you/the developer cannot login with that account).

Apply hardening on the system so that for instance spawning processes from Office Application or WMI (well-known malware techniques) are blocked.
This is also known as Attack Surface Reduction.

Enable monitoring software to help you identity suspicious behavior, linked with direct feedback to the developer.
The Endpoint Detection and Response tools can also notify you of suspicious actions.

Establish priority for security warnings on developer machines and accounts in the SOC team, so alerts are investigated with high priority by a team of SOC analysts with developer knowledge.

But the most important of all is continuous training and awareness! </>

Erik Oppedijk
Cloud Architect, Public Speaker and Trainer

xpirit.com/erik

[2] https://www.theregister.com/2021/03/17/microsoft_vulns_admin_rights/