Stuxnet, a Weapon of Cyber Warfare and an Opportunity to Learn About Security in DevOps

George Malki, Mayuri Salunke, Hasan Kalzi

May 2022

Geomal@kth.se, salunke@kth.se,
Kalzi@kth.se

Introduction

Cyber attacks can be defined as “An act in cyberspace that could reasonably be expected to cause harm”[21]. Cyberwarfare is when one nation does a cyber attack on another intending to create havoc on government and civil infrastructure. However, it’s hard to find real-life examples of such attacks, since no direct loss of life can be witnessed and no nation will willingly take responsibility for an attack on a foreign country. And one such example that we will be discussing is a cyber-weapon that has come to become one of the most infamous cyber warfare attacks that a nation has used against another, known as the Stuxnet.

The success of Stuxnet has fueled a cyber-weapon arms-race within governments and organizations. This in turn affects the end-users, including the general population. Proving that the need for secure software development methods is urgent. This can be achieved by adopting and implementing DevSecOps practices such as “Practice Secure Coding” and “Embrace Automation” and a few others that we will be exploring in this essay.

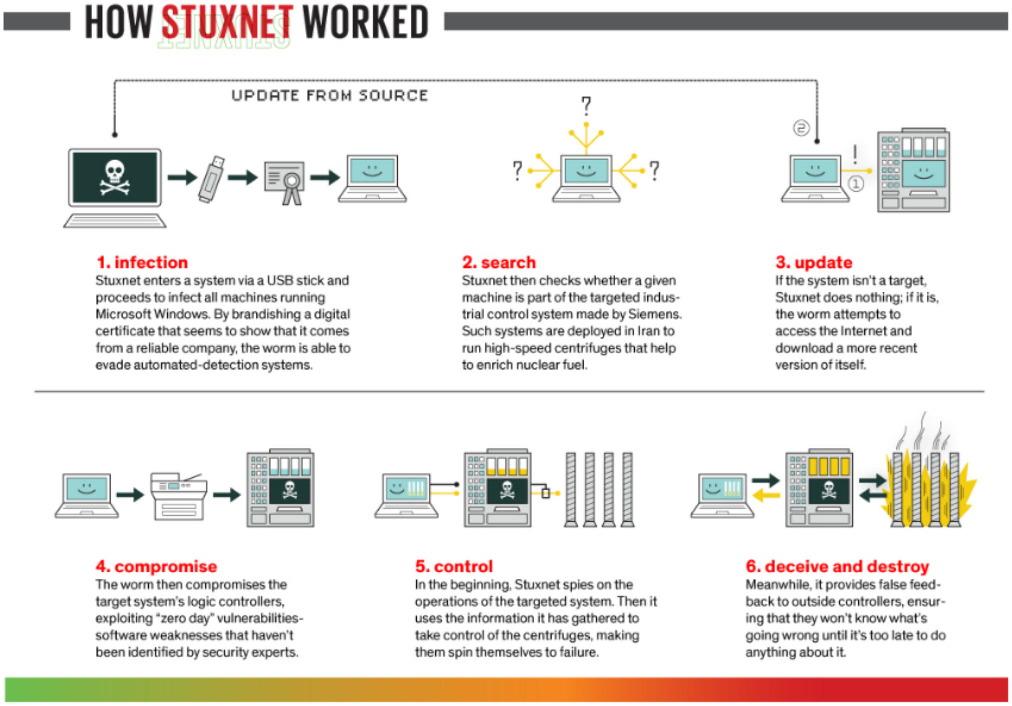
Background

The Stuxnet Worm

Stuxnet worm was first discovered by a researcher for a Belarusian cyber security company VirusBlokAda in June of 2010[9]. The malware instantly drew a lot of attention, not only for its sophistication but also for its use of four previously undiscovered zero-day exploits which was unheard of before[16]. Unlike many pieces of malware that were mainly designed and used for espionage nations used to spy on each other, the makers of Stuxnet had another goal in mind. Stuxnet was designed and built with the intention to sabotage centrifuges in the power facilities of Natanz in Iran. It reportedly destroyed numerous centrifuges by causing them to burn themselves out and over time, the virus was modified to target other facilities like water treatment plants, power plants, and gas lines[14].

Stuxnet was a multi-part worm that traveled through USBs and was spread through Microsoft Windows computers. It would search the infected computers for signs of Siemens Step 7 software (an engineering tool used for configuring and programming controllers) and would update its code over the internet and send damage-inducing instructions to the PC-controlled equipment. At the same time, the virus sent false feedback to the main controller and hence, anyone monitoring the equipment would not realize the presence of any problem until the equipment self-destructs[14].

Stuxnet was one of the first cyberwar weapons that were discovered which were able to target and destroy physical infrastructure. And by legitimizing such cyber attacks, an opportunity for a significant blowback has been created[17]. Instead of treating cyberspace as an opportunity for innovation and exchanging information, Stuxnet has now opened the realms of cyberspace for more cyberwars. It has now become a muse for cyber attackers to achieve in order to create a better virus. Hence, in the near future, the cyberwar race will be more difficult to ameliorate than the race of nuclear weapons.



1. infection
Stuxnet enters a system via a USB stick and proceeds to infect all machines running Microsoft Windows. By brandishing a digital certificate that seems to show that it comes from a reliable company, the worm is able to evade automated-detection systems.

2. search
Stuxnet then checks whether a given machine is part of the targeted industrial control system made by Siemens. Such systems are deployed in Iran to run high-speed centrifuges that help to enrich nuclear fuel.

3. update
If the system isn't a target, Stuxnet does nothing; if it is, the worm attempts to access the Internet and download a more recent version of itself.

4. compromise
The worm then compromises the target system's logic controllers, exploiting "zero day" vulnerabilities-software weaknesses that haven't been identified by security experts.

5. control
In the beginning, Stuxnet spies on the operations of the targeted system. Then it uses the information it has gathered to take control of the centrifuges, making them spin themselves to failure.

6. deceive and destroy
Meanwhile, it provides false feedback to outside controllers, ensuring that they won't know what's going wrong until it's too late to do anything about it.

Figure 1: Step-by-Step explaination of how to stuxnet attack took place, DAVID KUSHNER 2013 Image source

Zero-Day Vulnerability

A zero-day vulnerability in simple terms is a flaw[11]. It is an unknown vulnerability in the software or hardware of the system that when exploited can allow attackers to gain unauthorized access, allowing them to damage or exploit the system before someone realizes that something went wrong. It is named 'zero-day' as the attack happens once the attacker detects the flaw and exploits it by releasing malware, thereby giving no chance for the developer to create a patch to fix the vulnerability. And a clear case of this is the Zero-day exploits that were found in the Microsoft operative system Windows used in Stuxnet. Stuxnet did not exploit one, but four different zero-day vulnerabilities to delve and spread into the Microsoft Windows operating system. It used a Windows rootkit (collection of software which provides unauthorized access to a computer or an area of the software), antivirus evasion techniques, stolen certificates from trusted CAs, and peer-to-peer updates[15]. The stolen certificates from the trusted CAs were used for the malware to appear legitimate and to avoid detection by the traditional intrusion detection systems (IDS). While the peer-to-peer communication enabled the malware to self-update even when the compromised device had no direct access to the internet.

Supply chain attack

The Supply Chain Attack is an attack that targets third-party vendors who provide the vital services integral to the supply chain. They target developers or providers with access to your system or data to infiltrate the system. The attacker slips a malicious code or even a component into a trusted software or hardware that can hijack the application to turn any application the vendors sell or even an update they push out into a trojan horse. Stuxnet could be classified as a Supply Chain Attack since it has some of the tell-tale signs of such an attack. For instance, it relied on infrastructure misconfiguration as well as using stolen secret certificates to authenticate itself as one of the supply-chain programs that the centrifuges in the Uranium enrichment facilities relied on[7]. It is also noteworthy to mention that the Stuxnet virus was introduced to the target via an infected

USB stick[10], which is how it was able to be transferred into an air-gapped environment.

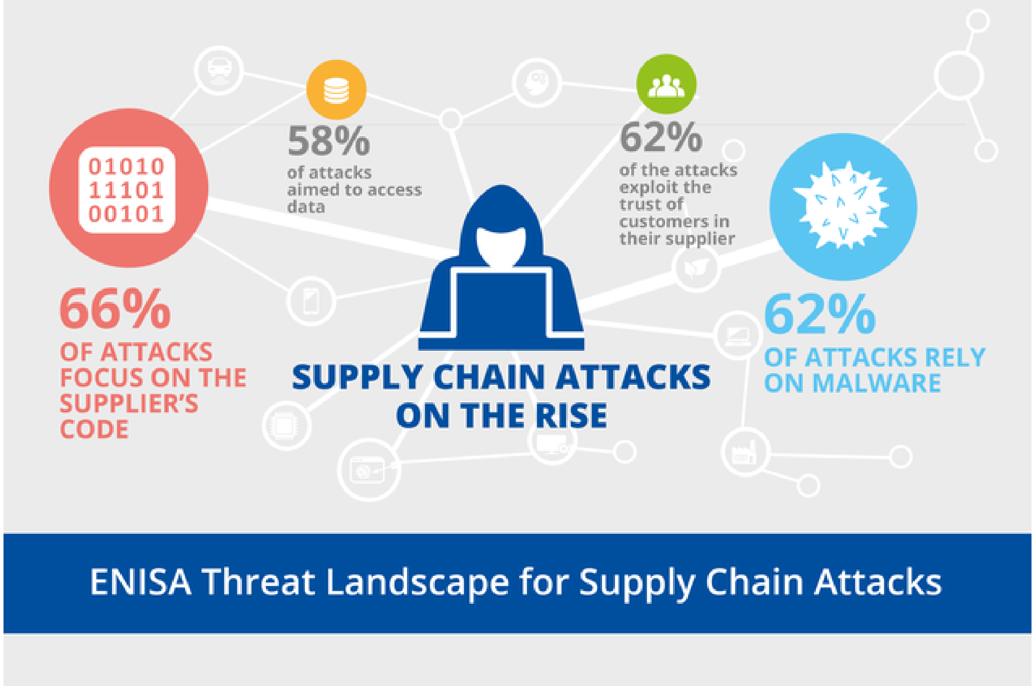

01010
11101
00101
66%
OF ATTACKS FOCUS ON THE SUPPLIER'S CODE
58%
of attacks aimed to access data
62%
of the attacks exploit the trust of customers in their supplier
62%
OF ATTACKS RELY ON MALWARE
SUPPLY CHAIN ATTACKS ON THE RISE
ENISA Threat Landscape for Supply Chain Attacks

Figure 2: statistics for Supply Chain Attacks, enisa europe 2021 Image source

Air Gapped System

This leads us to the concept of Air gapping. Air gapping or air walling in its simplest form is the act of physically isolating a single, or multiple computers from unsecured networks and is one of the most common security measures that is employed around the world. It disconnects the computer from any other computers or networks[12]. So how was Stuxnet able to install itself on an Air-gapped system anyways? While it's common knowledge that you shouldn't connect just any USB-thumb drive to any of your computers, let alone the computer that is used to control the centrifuges that enrich and create military-grade Uranium, at least one Iranian engineer didn't get the memo. The engineer bridged the local and secure network with the outside world for a brief moment allowing the worm to install itself on the device by inserting the USB drive into the computer.

DevSecOps

Without adequate involvement of a security team, deploying a piece of software, whether be software changes or new additions, is more likely to contain vulnerabilities, some of which could be catastrophic. As a result, the collaboration between the development teams inside an organization and security teams has gained popularity in the last couple of years[5]. According to a research paper[20], DevSecOps is "the concept of integrating security principles through increased collaboration between the development teams, operations teams, and security teams of a DevOps organization." DevSecOps stands for development, security, and operations. Its goal is to make automation, culture, and platform design integrate security collectively throughout the IT lifecycle[2]. Using DevSecOps has many advantages like enhanced speed and agility for security teams, lowering costs, and free time for continuous improvement[18]. DevSecOps contains best practices such as:

1. **Practice Secure Coding:** While it is reasonable to assume that secure coding is important to ensure that the software is resistant to vulnerabilities, it's important

to remember that everyone makes mistakes, including developers. Coding errors caused by developers might invite some unnecessary risks. Therefore, it is important to invest in training developers on secure coding even if it translates to a high time and cost investment. This can for example be done by establishing secure-coding standards that the developers can adhere to in order to write secure code.

2. **Embrace Automation:** The usefulness of leveraging automation whenever possible is as relevant in DevSecOps as it is in DevOps. Automation in DevOps has allowed the development pace to increase dramatically, therefore it is equally important to Automate security to ensure that the development process is not slowed down. Careful and thoughtful planning is crucial when automating security testing.

   Different tools fulfill different purposes throughout the development process. For example, Static application security testing (SAST) tools are meant to scan changes in the source code in order to identify sources of vulnerabilities. Scanning daily changes should be prioritized over scanning the entire application daily since it's more efficient and easier to catch and patch vulnerabilities[1]. Dynamic application security testing (DAST) on the other hand can help users find vulnerabilities during runtime through simulated attacks[13]. Open Web Application Security Project (OWASP) has lists of known vulnerabilities that a DAST and SAST can check for in the application[8].

3. **Shift Left:** Shift-left testing refers to focusing on security very early on in the development process instead of waiting until the last stages of the development[6]. It builds upon the idea that the earlier we focus on security the earlier we can identify vulnerabilities and the easier and cheaper they are to fix.

4. **Container Scanning:** Application development and deployment on containers are growing in popularity[3]. Container scanning is the process of scanning containers and their components to identify potential security threats. A container scanner is an automated tool that analyses the various components of the container to detect any security vulnerabilities.

5. **Monitoring:** When a product is deployed and exposed to the outside world, monitoring becomes an essential part of DevSecOps. Organizations need to monitor the product for any outside attacks or vulnerabilities that could cause leaks. This can be done via runtime application self-protection (RASP) tools which automatically detect and block computer attacks in real-time[4].

Discussion

Security is of utmost importance for critical infrastructures such as power generation, water treatment, electricity production, telecommunications, and other platforms that are critical for a country. Having such a high value makes critical infrastructures an extremely attractive target for foreign adversaries such as riveling countries and/or cyber-terrorists. Stuxnet isn't the first cyber-weapon that has been used to cause havoc on a country's infrastructure, nor will it be the last. The attacks that were used by Stuxnet, while sophisticated, have become more common, especially Supply-Chain Attacks which rely on the distributed nature of modern software development[19]. In our opinion, the techniques that are used in DevSecOps could've hindered, if not prevented, some of the attacks that Stuxnet exploited.

Stuxnet's Zero-Day attack relied on four undetected vulnerabilities in Microsoft's operative system Windows. We're confident that the developers in Mircosoft used some form of "early DevOps" while developing Windows. It's also reasonable to assume that the developers of Windows know a thing or two about secure coding practice and left-shift so we don't think that it would have prevented these vulnerabilities. Though we are fairly sure that the automated security testing tools that we currently use are a lot better/more effective than the ones that existed in that time era (if they existed at all since we weren't able to find which version of Windows was infected). These vulnerabilities would've most likely been detected had automatic testing tools, such as SAST been used during the development of Windows. SAST can be used earlier on during the development cycle since it can scan the source code or binary as soon as the code is deemed feature-complete without the need to execute the application. DAST on the other hand can simulate known attacks on the application during run-time. But the reason why DAST can't really be used in this case is that DAST is more Targeted towards Web-applications[13]. Another major benefit of the current technology is the existence of the Open Web Application Security Project which makes detecting a broader range of vulnerabilities easier. It's possible to automate SAST tools to check for these vulnerabilities[8]. The reason why Stuxnet was able to evade the monitoring system that was in place is that Stuxnet had the ability to send false feedback to the main controller to avoid setting off any alarms.

Supply Chain Attacks on the other hand are a bit more complicated to protect against since an organization could have code that is vulnerability free and still be a victim of a Supply Chain Attack. The reason is that the vulnerability can reside in one of the components that the program has a dependency on. DevSecOps can help protect us against such attacks thanks to automated security testing tools such as SAST, DAST, and container scanning. By implementing these techniques we increase visibility over potentially insecure third-party code during the development cycle.

Further, none of the aforementioned best DevSecOps practices could prevent the bypassing of the Air-Gapped System. In cases like this, it is not the DevSecOps practices, but rather the employee's awareness of security practices that will prevent such incidents from occurring.

Conclusion

With the advancements that mankind has been able to achieve in the last couple of decades, we have entered a generation of warfare, Cyberwarfare. Though on paper it doesn't sound as deadly as the typical warfare, it has the same catastrophic potential as old-school warfare without the need to shed a single drop of blood. Cyberwarfare can come in many shapes and forms but the underlying goal is the same, to disrupt the infrastructure of the other nation to cause damage. And through this essay, we have explored how adopting and implementing DevSecOps practices can help protect and prevent such attacks. Upon researching and understanding the vulnerabilities exploited by Stuxnet, we believe that these vulnerabilities could have been detected and patched using the security tools mentioned in the essay. Hence, DevSecOps is an integral part that needs to be integrated into every phase of DevOps to prevent such attacks from occurring.

References

[1] 5 devsecops best practices you must implement to succeed. https://www.tigera.io/learn/guides/devsecops/devsecops-best-practices/. Accessed: 2022-05-30.

[2] Devsecops guide. https://tech.gsa.gov/guides/dev_sec_ops_guide/. Accessed: 2022-05-30.

[3] Everything you need to know about container scanning. https://snyk.io/learn/container-security/container-scanning/. Accessed: 2022-05-30.

[4] Rasp security. https://www.contrastsecurity.com/knowledge-hub/glossary/rasp-security. Accessed: 2022-05-30.

[5] The rise of devsecops. https://www.devopsonline.co.uk/the-rise-of-devsecops/. Accessed: 2022-05-30.

[6] Shift left. https://devopedia.org/shift-left. Accessed: 2022-05-30.

[7] Software supply chain-still a vulnerability for our critical infrastructure. https://blubracket.com/software-supply-chain-still-a-vulnerability-for-our-critical-infrastructure/. Accessed: 2022-05-30.

[8] Source code analysis tools. https://owasp.org/www-community/Source_Code_Analysis_Tools. Accessed: 2022-05-30.

[9] Supply chain attacks. https://docs.microsoft.com/en-us/microsoft-365/security/intelligence/supply-chain-malware?view=o365-worldwide. Accessed: 2022-05-25.

[10] Unprecedented look at stuxnet, the world's first digital weapon. https://www.wired.com/2014/11/countdown-to-zero-day-stuxnet/. Accessed: 2022-05-30.

[11] What is a zero-day exploit? https://www.fireeye.com/current-threats/what-is-a-zero-day-exploit.html. Accessed: 2022-05-25.

[12] What is an 'air gap' in computer security and networking? https://history-computer.com/air-gap/. Accessed: 2022-05-30.

[13] What is dynamic application security testing (dast)? https://www.microfocus.com/en-us/what-is/dast. Accessed: 2022-05-30.

[14] What is stuxnet? https://www.trellix.com/en-us/security-awareness/ransomware/what-is-stuxnet.html. Accessed: 2022-05-25.

[15] Reihaneh Safavi-Naini Alvaro A.Cárdenas. Chapter 25 - security and privacy in the smart grid. In Sajal K. Das, Krishna Kant, and Nan Zhang, editors, *Handbook on Securing Cyber-Physical Critical Infrastructure*, pages 637–654. Morgan Kaufmann, Boston, 2012.

[16] Marie Baezner and Patrice Robin. Stuxnet. Technical report, ETH Zurich, 2017. Accessed: 2022-05-25.

[17] Sascha Meinrath Jeff Landale. Opinion: The troubling stuxnet effect. *csmonitor*, November 2015. Accessed: 2022-05-25.

[18] Sumo Logic. What is devsecops? May 2019. Accessed: 2022-05-30.

[19] N/A. 2021 software supply chain security report. Technical report, Argon Security, 2021. Accessed: 2022-05-25.

[20] Akond Ashfaque Ur Rahman and Laurie Williams. Software security in devops: Synthesizing practitioners' perceptions and practices. In *2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED)*, pages 70–76. IEEE, 2016. Accessed: 2022-05-25.

[21] Michael Robinson, Kevin Jones, and Helge Janicke. Cyber warfare: Issues and challenges. *Computers Security*, 49:70–94, 2015.