Christopher Rucinski
Developing Secure Software
SE-4930
Podcast Summary 1

Stuxnet Under the Microscope

In a podcast with Security Expert and CTO of Cigital, Gary McGraw – lead by news editor Lee Gaber, he discussed how Stuxnet works, what it does, what makes it interesting, what type of systems it attacks, how this malware is different from other malware, and what this means for the future.

“Last year marked a turning point in the history of cybersecurity—the arrival of the first cyberwarfare weapon ever, known as Stuxnet. Not only was Stuxnet much more complex than any other piece of malware seen before, it also followed a completely new approach that's no longer aligned with conventional confidentiality, integrity, and availability thinking. Contrary to initial belief, Stuxnet wasn't about industrial espionage: it didn't steal, manipulate, or erase information. Rather, Stuxnet's goal was to physically destroy a military target—not just metaphorically, but literally” (Langner, 2011).

As by McGraw, Stuxnet was a targeted information warfare attack that was aimed at Iran, and its nuclear centrifuges systems. Stuxnet had a delivery mechanism that consisted of several Microsoft Windows zero-days, and a couple of stolen keys (whatever that means); however, this virus had a specific target in mind – this virus was looking at only targeting Siemens industrial software and equipment, specifically their process-level controllers (PLCs). There has been evidence that Stuxnet was successful in its goal to mess up the physical processes of the PLCs and caused physical damage of the centrifuges that were connected to the PLCs at the Natanz nuclear facilities in Iran.

This malware spread due to several different aspects. First, and foremost, the PC's at the nuclear facility were Windows-based, and was not updated. Secondly, the malware was introduced to the system by an “integrator” through the use of a USB stick – most likely unaware of the malware on the USB stick. Lastly, the malware could now search for the intended target through Peer-to-Peer communication until it was able to find the PLCs. This malware had several safe-guards built-in, and it always made sure that it only set in motion its destruction if particular instantiations of particular PLCs where detected.

McGraw states that Stuxnet's payload, or the actions the malware took beyond simply infecting files, was DLL inter-positioning (or placing in DLL that gets in between or intervenes messaging), and overriding several methods.

After Lee Gaber asked McGraw what type of general vulnerablilites did Stuxnet take advantage of, McGraw got more in depth on the delivery mechanisms used, the payload, and the attack code. He stated that the originators of the code base had four zero-days to work with. Zero-days are security exploits in a software system that the developers don't even know about. However, it is also known that Stuxnet also used an addition security vulnerability that was already known about by Microsoft. As a side not, only two of the zero-days have been patched, and I have not heard if any of the others have been.

Stuxnet did not go after any bugs in Siemens code base; it went after the design of the PLCs, along with DLL. Targeting the DLL – that was made by Siemens – from the PLCs were made easy due to the fact that they where not protected by cryptography, stripped from symbol tables, or signed. They where not stealthy, and had no protection mechanisms in place.

The attack code payload was well constructed and very sophisticated. It used general root-kit ideas of hiding itself from users. If a user of the nuclear facility at Natanz wanted to query the PLCs for registry

values after the PLC was infected, the result that was sent back to the user would be false values, but values that indicated that everything was working normally. Due to the sophistication of Stuxnet, McGraw believes that a "nation-state" could be the only entity that could have built this, but also believes that it was created with a loose affiliations of contractors around the world.

From here on, McGraw talks about the implications of Stuxnet on industrial components, and the security field. He starts off my stating that the delivery mechanism in Stuxnet was sloppy and had a "blowback" effect on non-industrial PCs. Because of this, all major anti-virus companies have built detection systems to guard against Stuxnet for regular consumers; however, the users of PLCs, such as those in nuclear facilities, are out of luck getting support like regular consumers get due to the much lower concentration of users. They were able to consult with several major anti-virus companies to fix their problems individual though. He continues and states that industrial control companies have not thought a lot about what to do when their PLCs get infected, and that is a problem, because shutting down the system is not always possible.

So what can be done to help prevent these type of attacks. Companies need to build contingency plans, and come up with better engineering that is harder to attack from threats such as Stuxnet. However, a lot of systems are infected because they started off broken from design. This is because in the early days, the designers of those systems did not know that all these computers and components would all be interconnected through the Internet. Without that knowledge, the architects of the System did not have to worry about outside factors as much as architectures of software systems have to today. He brings up a point that the people building the Smart grid in the US will have to take lessons learned from Stuxnet and implement them into the design.

Since a good amount of software will rely on third-party vendors, due to time-to-market, and cost savings, he tells us that the developers have to ask those venders if they have any security engineering in their, and they must show you evidence of this. Ask for architecture risk analysis documents, threat models, and code reviews by static analysis just to name a few. This will make those vendors rethink what they put into the development of their products, because if they don't they will lose potential customers. Finally, McGraw, states that security requirements absolutely need to be treated the same as performance or robustness requirements.

From my readings online from the past year, I was able to find out a lot of information about Stuxnet and how it spread; however, there seem to be some speculation from highly notable sources that Stuxnet was not a "one-man army" as it possible had help from at least four other pieces of malware. So my question deals with how did Stuxnet and these other pieces of malware work together? Is Stuxnet done, or just laying dormant until the next target is determined?

This podcast described aspects of Stuxnet that I had not known about previously, and allowed me to find more detailed sources of information about this interesting topic. I had learned more in depth about Stuxnet's payload, the delivery mechanisms that were in place, and how they were developed. While understanding about Stuxnet more in depth was interesting, that is not what will allow me to write better software – not directly at least, it was the information at the end of the podcast that will be beneficial. Thinking about software design not in the present, but also in the future, and asking third-party venders to show you proof that their software is secure by security documents upfront.

Christopher Rucinski
Developing Secure Software
SE-4930
Podcast Summary 1

Work Cited

Langner, R. (2011). Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Computer Society*, *9*(3), 49-51. Retrieved from http://www.computer.org/csdl/mags/sp/2011/03/msp2011030049.html

McGraw, G. (Performer), & Garber, L. (Editor) (2011). *Gary mcgraw explains stuxnet* [Web]. Retrieved from http://www.computer.org/portal/web/computingnow/stuxnet-under-the-microscope

The world of industrial control is ripe for exploits like this, as many of the systems are very crude and may not have been solidly engineered. But, now that they are on the network, security becoems very important.