Slammer Worm and David-Besse Nuclear Plant

Michael Holloway
July 16, 2015

Submitted as coursework for PH241, Stanford University, Winter 2015

Slammer Worm Background

Fig. 1: An image of the David-Besse nuclear plant in Ohio. (Source: Wikimedia Commons)

The Stuxnet Worm first became a significant internet security threat in 2003. [1] The worm itself is known by several names including SQLSlam, Slammer, and Sapphire. It was a network worm that spread through computer systems, exclusively in memory. [1] The worm itself was remarkably only 400 bytes long. Slammer infected process spaces of Microsoft SQL servers. [1] The worm relied on the common hacking tactic of buffer overflow. Once it had penetrated the SQL server, it continued to run in an infinite loop on that server. [2] Slammer also used each server it had penetrated as a port by which it would send copies of itself to other random IP addresses. [1] The worm would not stop sending copies of itself to other servers until a user at the original port noticed the existence of something strange, and halted all processes on that server. [1] It was said that at the time the Slammer worm was the fastest spreading worm of all time. [2] Many experts calculate that the worm was actually capable of crashing the entire Internet within fifteen minutes of its release. A majority of the effected SQL servers belonged to corporate computer systems. The worm used great amounts of CPU power and energy in order to continue replicating and transmitting itself to other computing systems. [2]

David-Besse Nuclear Plant

One of the greatest effects of the Slammer worm, which wreaked havoc worldwide by clogging Microsoft servers, occurred at a nuclear plan in Ohio in 2003. [3] The worm first embedded itself into a David-Besse contractor's computer which allowed it to proceeded to access the David-Besse corporate network. An image of this nuclear plant is shown in Fig. 1. [4] Once in the corporate network the worm found its way into the reactor's processing control systems because the processing control system was linked to the public corporate network. [4] The worm froze the employees of the reactor facility out of the Safety Parameter Display System that delivered "crucial safety indicators ... like coolant systems ... and external radiation sensors." [4] Because of the reactors lack of separation to a public network, the slammer worm was able to penetrate and cause harm to the reactor's internal functions. As a result, the worm "disabled a safety monitoring system for nearly five hours." [4] All of the employees at the Ohio plant were unable to access the Safety Parameter Display System. This system was responsible for monitoring the most important "safety indicators at [the David-Besse] plant." [4] For example, employees were unable to monitor the core temperature sensors at the plant, a crucial safety hazard at a nuclear energy plant.

Implications of the Slammer Worm Incident

Many experts claim that the Slammer worm incident at the David-Besse nuclear plant illustrates a serious problem that continues to exist at nuclear energy facilities around the United States. [2] The issue at hand is that there is a drastic lack of cyber security protection in place at nuclear facilities. For example, the David-Besse plant did have a functioning firewall; however, the outside contractor established an internet connection behind this firewall linking his own companies network directly to the plants computing system. As a result, the slammer worm was able to bypass the David- Besse firewall and wreak havoc on their computing systems. [2] At the time of the incident, the Nuclear Regulatory Commission did not restrict remote access to nuclear facilities which posed a major security threat as seen above. [2]

References

[1] D. Moore et al., "Inside the Slammer Worm," IEEE Secur. Priv. **11**, 33 (20113).

[2] P. Boutin, "Slammed!," Wired Magazine, July 2013.

[3] W. Knight, "Slammer Worm Chokes the Internet," New Scientist, 27 Jan 03

[4] T. L. Hardy, *Software and System Safety: Accidents, Incidents, and Lessons Learned* (AuthorHouse, 2012).