Intrusion Detection Systems

Innovations in Software Security

Thomas Travers
Grantham University
CS406

Innovations in Software Security

1. Browse the Internet and select a story about an innovation related to software security.

One of my favorite innovations to software security is Intrusion Detection and Intrusion Prevention Systems firewall policies. According to threatstack.com, the initial model for the earliest IDS is a result of a report published by the US Air Force in the 1980's. During this time period, there was a shift to networked information technologies and it was a rapid growing industry in both government and civilian use. Governments and banks were among the first largest adopters, and many large companies adopted this tech as well. With the rise of this technology, there was a need to monitor and control access to information systems to prevent unauthorized use, data access, or sabotage.

2. Research to answer any questions you have about the story.

Most of my questions were about the origin and inspiration behind IDS and IPS systems. The article itself adequately answered all of my questions.

3. Write an article about the story. Include a link to the story you chose. Use APA style; cite sources and include references.

During my time running self hosted web services for the majority of this last decade, and hosted services for almost a decade prior, I have ran into a few road bumps along the way and learned a lot about server administration and security. Some of the biggest challenges I faced was utilizing the proper tools in order to collect analytical data to determine where my bottle necks and vulnerabilities are. Among the first thing I discovered is resource monitoring. I found that I constantly began getting resource drain every now and then. I gathered research material and formulated a plan to scale. I learned to use monitoring tools to gauge when to scale and a formulated a plan how to scale. Since this is web services, most of my scaling included load balancing, but sometime required adding RAM. When doing this for servers that produced little to no income, it became not very cost effective to maintain. The solution here was to self host instead of using a VPS. The benefit of a VPS is that it was behind the hosting services firewall regardless of the firewall used on the VPS itself. The hosting service provided protection from brute force as well as adequate and basic IDS/IPS capabilities. Once my services were ported to a self hosted platform, I started having resource drains again. This time, I began examining logs to determine what the problem was. I realized that there were multiple failed login attempts by same IP addresses attempting to access services such as SSH, FTP, CUPS, and Telnet (which was not enabled). The server utilizes system resources in order to validate or invalidate these logins. And the more attempts these logins make that fail, the more data the intruders collect about what isn't the password. In my research, I concluded that these were *brute force attacks. This is a type of attack where the user employed a script that attempts to guess the password in an organized fashion until access is granted. Such as the first attempt being a, the second attempt aa, then ab, then ac, then ad, then ae and so on until you crack the password. The client was sending a few attempts per second while looking for the password many of them seemed simultanious in the log time stamps telling me that it was an automated process. The method I ended up using to thwart this specific attack was to employ a firewall rule with IP Tables in order to stop the attack. I then set up a system to easily manage firewall rules using ufw. Ufw made IP tables more human readable and generated easy to read reports including comments that allowed me to identify blocked attacker IP's. Some of the issues I still had at this point was*

(a) *the system was not automated. This means I still manually had to examine logs and apply the ban manually with ufw.*

(b) *Much of the public internet devices use dynamic IP addresses. This means I can block them from service from my server, but the abuser will get a DHCP release and have a new IP in a couple of days.*

(c) *Another issue with dynamic IP addresses is that the next person to be assigned the IP address used by the abuser will not be able to access my web services.*

The solution to this was an IDS / IPS firewall system.

In 1980, James P. Anderson of the U.S. Air Force and a member of the Defense Science Board Task Force on Computer Security issued a reported called "Computer Security Threat Monitoring and Surveillance" which is often credited with introducing automated IDS techniques. Not long after the release of "Computer Security Threat Monitoring and Surveillance", the first rule based IDS model was developed. During the 1990's, many more government and civilian facilities employed network information systems into their business models. Intrusion detection models and techniques evolved to detect increasingly more complex attacks leading to developing and utilizing more complex models and analytical techniques including pattern analysis and techniques for unknown threat detection. Much of this was modeled after existing Antivirus techniques of the time (Team, 2020).

I started with fail2ban on my server. Fail2ban is excellent at providing brute force protection. It is a rule based IDS / IPS that analyzes log files for patterns that match a set of rules. It then enforces a policy that the administrator sets forth based on the rules that it matches. For example, my rule for SSH intrusion detection examined the log files for failed SSH login attempts on port 22. If the same IP address failed 3 times, then it applied a ban action. The custom action utilized ufw and created a rule that bans the IP address for a period of two weeks. We don't want the ban permanently because the IP address will cycle. The server also uses SSH key to login in order to increase complexity and this key is used by the admin's user account only. Root login is not allowed on any of my server systems further increasing login complexity making it harder to brute force as they require an username capable of login. Fail2ban is a rule based IDS / IPS system employed. I ended up setting up another IPS / IDS system to work with fail2ban. I found snort. Snort analyzed network packets and look for patterns to

(a) *Identify known patterns as malicious*

(b) *Identify normal patterns to separate unknown patterns and flag them as anomalies or potential threats (unknown threat detection from non-normal network patterns).*

The ufw rules used in the enforcement of IPS workflow allow me to easily “unban” an IP that was accidentally ban instantly and in real time in a more human readable fashion. My system is now completely automated and this saves valuable resources from the computer attempting to Authenticate every failed login attempt. I even have rules that look for IP address sequences for scripts that cycle IP addresses in order to thwart off IDS / IPS firwalls. I now spend my time looking at network bandwidth, and resource usage on my server screen and take a peek through log files every now and then as well as logged in users to ensure none of the brute force attacks were successful. Another good practice is that every two quarters, I swap out my SSH key further increasing the complexity required to break into the system. The more complex my login system is, the harder the abuser has to work in order to breach.

4. Upload your article to the dropbox, and share your article with the class on the discussion forum.

Sources Cited:

Team, T. (2020, February 24). The History of Intrusion Detection Systems (IDS) - Part 1. Retrieved August 02, 2020, from https://www.threatstack.com/blog/the-history-of-intrusion-detection-systems-ids-part-1