IdealPrevention, an Intrusion Prevention System by Atomhid

Programmer/Author: Daniel Alexander Apatiga

Version: 2.0.1

What exactly is an intrusion prevention system?

Atomhid's free process firewall and intrusion prevention system can prevent malicious software introduced either in person, or activated remotely on a client device by a hacker, and it can stop it within a reasonable amount of time, ideally. Unlike other security software, the firewall for this is coded differently: instead of relying on malicious-code signature detection and heuristic analysis of applications of which often times is innaccurate, Atomhid's I.P.S. stops the process, and this from a security standpoint is not so bad. Should an ideal security system stop malware from communication with a server somewhere but have an insufficient and breakable firewall, and allow permission-granted software susceptibility to man-in-the-middle attacks and code-injection? Or would you rather deploy a reliable firewall that will not guarantee malware from taking hold of a system, like Norton Security? Ideally, you would want both forms of security.

As a supplementary security application, this software only allows certain software on a list to operate on an operating system (o.s.) if and only if it accesses the session's layer; and, it can do it on minimal installations of Linux kernels, too. Once an unknown malware passes by typical security-software scans and its passive detection algorithms, when it is detected by Atomhid's i.p.s., the malware's process and sub processes will no longer function or if it runs again, it will be stopped, but only if it is not on the exemption list. An added benefit of using the GUI-less version of Atomhid's i.p.s. is that if you are on a minimal installation of a Linux kernel, this software can run and be configured to work in the background.

How Atomhid's i.p.s. works

By common computer-nerd nomenclature—an ideal intrusion prevention system prevents unauthorized users, or hackers, from accessing, modifying, deleting, and reading your files without your knowledge—but, the malware first must access the network before an i.p.s. can defend. In the meantime, before malware accesses the web/lan/wlan, malware can still damage your data and "prepare" it for later acquisition/confiscation/modification; that is the biggest weakness of this freeware. So, now that you are warned, here are the pros of this software. Malware includes: ransomware, rootkits, keyloggers (a sub-classification of spyware,) and varying types of viruses, those of which can take control over the application layer of the o.s.i. seven-layer model of a vulnerable network, and even change behavior of the underlying o.s. Atomhid's free process-firewall system is not intended as a replacement from buying an expensive router with firmware that includes an i.p.s., but it may do just that. *IdealPrevention* adds an additional layer of protection just in case malware communicates with the w.l.a.n/internet, especially since most malware generally uses the network to disguise itself well enough. The prevention of further mal-logic from happening is exactly what Atomhid's i.p.s. promises, but on a higher application layer instead of the link layer where traditional router-based i.p.s.' detect and defend against these attacks.

Since hackers, nowadays, have access to many hacker kits with preassembled malicious code that can be easily modified, and thus, they can creep stealthily by without detection by the most sophisticated security system. So, it is the job of the intrusion prevention system to stop it and report it to the system administrator: you; this software will do exactly that. Once the malware starts its process, if it is already stopped once, it is immediately stopped again, and again, etc. Malware used by the hacker would have a tough time accessing a server from where the hacker can send malicious commands to the host, or your device.

Of the many malware languets, or types of malware, the system administrator must prevent any of them from attacking vulnerable data. But, do you ever notice that in the most well-defended systems, security holes are easily punched through? On encrypted databases, news sources often narrate stories about data breaches happening to top companies, like Google; do you know what causes these breaches usually? Usually, it is a trusted person who happens to be at the site, but also, it can be a man-in-the-middle making the attack. In these nightmarish scenarios, failure is often the result of malware that was not stopped, because it did not fit any known malicious code signature.

A possible success scenario, as an example

Let’s say you are in control of an important project—all file permissions are correct, but you failed at preventing physical access to your computer’s terminal. A supposed friend uses your computer and gives himself permission to remotely access it with administrative priveleges, writes down your ipv4 address, and disables certain security protocols and alerts. This friend goes home to his computer and gains access to your computer through a program with known vulnerabilities of which also happens to be on a pre-approved list generated by your atypical computer security software, but he does not know that you are using Atomhid’s i.p.s. He already made sure that the program has access on the atypical security application's firewall running on the computer by giving, for instance, a SSH server complete access to the network. It has read, write, update, and delete control privileges rendering his attack undetectable by passive scans, and packet sniffing scans by the atypical security software cannot comprehend the encrypted commands since your "friend" is using the SSH protocol. Additionally, your "friend" installs additional malware that runs passively, waiting for further commands; this, he thinks, will go by undetected. Now, right before you are about to make that big presentation, of which your supposed “friend” wants to see fail utterly, this person sends a command to delete your project with administrative privileges to the remote server on your computer, but unbeknownst to your “friend,” he loses communication with the backdoor; you are able to have a great day, and your presentation goes well. Also, all data on your computer has not been accessed post intrusion, even it may have been prepared for sharing with the intruder. In effect, you were running Atomhid’s *IdealProtection*, which had succeeded at securing your computer by scanning the session layer for pre-approved software, and if an application was not on the list, it was stopped.

From the creator of *IdealPrevention* himself

I recommend using this software as a substitution to routers that come with business-level security or computer security software that operate as aforementioned. And as always, two layers of protection is better than one, especially when your main security software becomes compromised.

Drawbacks

- When running, *IdealProtection* can use a lot of c.p.u. power, and thus, sort of slow down your system. It depends on how many parrallel processers your cores have, and of course, the # of cores, until this becomes noticeable. More cores and processers the better!

- Figuring out what background processes should run is no easy task. You have to name a process, since the scanning algorithm works by text-pattern matching named processes, of which must be exact, with your list.

How to install and run

Windows 64-bit (most versions)

1. Run the MSI for Windows 64-bit versions.
2. Right click on the shortcut created on your desktop. Select properties.
3. Click on the "compatibility" tab.
4. Make sure the "Run as administrator" checkbox is enabled (checked.)
5. Run in the background.
6. That's it!

Configuration

Creating a list of exempted processes:

1. In a blank notepad file, delineate a list of process names (executable binaries typically with the ".exe" extension after it) that may use the session layer with a comma between names. Capitalization matters. This looks like this:

   ```
   chrome.exe,Edge.exe,MySQLWorkbench.exe,
   ```

2. Make sure that no spaces exist between a comma and a letter.

3. Make sure that the list ends with a comma even though no application follows it.

Passing in program arguments

This can be done in either the command line interface or on the graphical user interface.

Windows

1. Right click on the shortcut created on your desktop. Select properties.
2. Type in any of these parameter flags after "IPS" to get the desired configuration:
   - "-m": This is the mode of the app. Options are: "file," for loading allowed apps from IPS.conf, or "cli" for just sending a comma-delimited list of app names"
   - "-Tc": This value is the maximum amount of background threads for calls made to the sessions layer. The higher the number, the more intensive the c.p.u. utilization.
   - "-ip": This is the value of the ipv4 address for sending log information.
   - "-port": This is the value of the port number.
   - "-allow": This is a list of applications that can use the sessions layer.

Example

```
IPS -m file -Tc 5 -ip "192.168.0.3" -port 5531 -allow
"chrome.exe,Edge.exe,node.exe,npm.exe,"
```