1/1/2020

Setting up and Comparing Different HIDS

By CyberApprentice

**Introduction**

For a very long time, I have gone without having a host-based intrusion detection system (HIDS). Even after starting my cybersecurity career in college nearly 4.5 years ago, I’ve yet to adopt it. HIDS are security applications with the goal of detecting threats to the system. There are multiple vendors available, but the following products will be set up and evaluated: OSSEC, Wazuh, Samhain and AIDE.

Initially, this writing was going to focus solely on OSSEC and evaluating it, but it is important for security professionals to be aware of other vendors, their capabilities, and comparing strengths and weaknesses. Different environments require different needs. Unfortunately, the only drawback is that paid HIDS will not be evaluated. Also, the four products listed are the only popular findings. During my search, there was a lot of confusion of HIDS vs. NIDS (Network-based intrusion detection system), vs. Log analysis.

**OSSEC**[[1]](#endnote-1)

OSSEC advertises to be more than just a HIDS. It is claimed to be platform that mixes aspects of HIDS, log monitoring, and security information and event management (SIEM).[[2]](#endnote-2) It may be deployed on Linux, Solaris, Windows, and Mac OS X platforms through agent or agentless implementations. Not only is it multi-platform, but can be integrated across different applications and technologies, such as: smtp, sms, syslog, and SIEM products. Its key features include file integrity checks; collect, analyze and correlate logs to alert on suspicious activity; rootkit detection; and incident response.

OSSEC architecture is compromised of the server, called the manager, agents, and agentless as shown below in *Figure 1*. The server stores all configurations and rules. It must allow UDP port 1514 for communicating with agents. Agents are forwarders that collect system information. They are resource efficient, low privilege, and chroot jail isolated. Agentless is a device that cannot have an agent installed on them (e.g. firewall, switch, or router) but is instead scanned for integrity checks. During the installation, the technician is given 4 options: server, agent, local and hybrid. Local is the equivalent of server without receiving external/remote logs (e.g. from agents). Hybrid is a combination of agent and local, so the server’s information would be included within the available analyzed files.

Figure 1 - OSSEC Architecture



To install, I will be using a Windows virtual machine (VM) running the Ubuntu app within the Windows store that allows Ubuntu and Windows to run concurrently—as this is how I would run it on my personal system. Unfortunately, wen OSSEC states that OSSEC may run on Windows, they are referring to the agent, not the server, so this is the optimal setup without having to have an appliance, VM, or any other mechanism that won’t have dedicated resources or being inconvenient to access.

Following the documentation, we do the following commands to go through the entire installation process. The latest release is at 3.5.0, however, the documentation doesn’t go past 3.3.0. Initially, I attempted to install 3.5.0. Throughout the process, I ran into some missing dependencies. It is unclear if this is because the limited Ubuntu sandbox that Windows provides, or if it this is a release issue. To test this, I conducted the same process as seen below and ran into the same issues. Nevertheless, *pcre2*, *libevent-dev*, *zlib1g-dev*, and *libssl-dev* were dependencies that were missing within the documentation that had to be manually downloaded. After installing everything and attempting to start the OSSEC server, I ran into the issues noted below:

Starting OSSEC HIDS v3.5.0...

ossec-analysisd: Configuration error. Exiting.

Viewing the log file, I receive the following:

ossec-analysisd(1450): ERROR: Syntax error on regex: '\(pam\_unix\)$': 9.

ossec-testrule(1202): ERROR: Configuration error at '/etc/decoder.xml'. Exiting.

This could potentially be linked to pcre2, as I downloaded the latest version, 10.34. I took the time to try the latest stable release (3.3.0) that matched the documentation. As stated, the same dependencies were missing, but after successfully installing them and the desired, preconfigured *pcre2* version (10.32) that was referenced, but not included, I was able to successfully start the process. The correct process is displayed below in

Table 1 - Installing OSSEC 3.3.0

|  |  |  |
| --- | --- | --- |
| Installing OSSEC version 3.3.0 on Ubuntu 18.04 WSL | | |
| Step | Description | Command(s) |
| 1 | Download dependencies | $ sudo apt update && sudo apt full-upgrade -y  $ sudo apt-get install build-essential libevent-dev zlib1g-dev libssl-dev  $ wget https://ftp.pcre.org/pub/pcre/pcre2-10.32.tar.gz  $ tar -zxvf pcre2\*.tar.gz |
| 2 | Download latest release and extract file | $ wget <https://github.com/ossec/ossec-hids/archive/3.3.0.tar.gz>  $ tar -zxvf 3.3.0.tar.gz  $ mv pcre2\* ossec-hids-3.3.0/src/external |
| 3 | Run script | $ cd ossec-hids-3.3.0  $ sudo ./install.sh  Parameters:  Language – English (default)  Installation type – server  Directory - /var/ossec (default)  Email notification – Yes  Email – [email] (no password required)  SMTP Server – (default found)  Integrity Check – Yes (default)  Rootkit Detection – Yes (default)  Active Response – Yes (default)  Firewall-drop Response – Yes (default) |
| 4 | Start | $ sudo /var/ossec/bin/ossec-control start  $ sudo /var/ossec/bin/ossec-control status |
| 5 | Installing Agents | For Windows, you just run the executable and follow the wizard  On the Server:  /var/ossec/bin/manage\_agents  Add agent  Extract key Copy key  On the Agent:  C:\Program Files (x86)\ossec-agent\ossec-agent.exe  Manager IP: 127.0.0.1  Authentication Key: <Copied string from the server> |
|  | To uninstall on Ubuntu | # /var/ossec/bin/ossec-control stop && rm -rf /var/ossec && rm /etc/init.d/\*ossec\* && rm /etc/ossec-init.conf |
|  | To uninstall on Windows | Go to the folder where OSSEC is located (C:\Program Files (x86)\ossec), run “uninstall.exe”, and follow the wizard |

The next step in the deployment is be configuring agents. Even though my installation of local would make sense since I’m only monitoring my system, but since I’m using the limited Ubuntu environment,

Resource utilization

Limits

Since I have a server and agent on the same system, I tried to test if I could use IPv6 link local addresses instead of IPv4 addresses. Link local addresses hop between links on the system. Because of this, I thought it would be faster. I was unsure how the packet flowed if it addressed itself. And there are no entry within the ARP table. After simulating the act of pinging one’s IP address within Cisco’s Packet Tracer, the packet never crossed the wire and the difference was marginal at best between the two protocols. Also, showing the Windows VM’s routing table shows that pinging one’s self and the link-local IPv6 address are both considered “On-Link”. However, the fact still remains that IPv6 was not working as an addressable host. I could ping the address, but OSSEC would only print the following log to show that it is trying to connect, but is failing:

ossec-agentd(4101): WARN: Waiting for server reply (not started). Tried: 'fe80::2'.

Personal Experience

In my experience setting up, reading, and using OSSEC, it was disappointing. A popular tool such as OSSEC carries high expectations. To begin with, the documentation was not as broken down and defined in comparison to other tools such as Wazuh. Also, it was outdated. The worst part was that it did not list or include the dependencies which wasted a lot of time when initially setting it up. The entire first experiment took at least an hour of pure hands-on-keyboard. After figuring out the correct steps unique to my environment, it takes around 5-10 minutes. For the future, I hope the level of detail that documentation covers, and necessary coverage improves, but I’ve come to realize that the development staff don’t seem to be compelled to do so. With that said, a future of OSSEC seems bleak.

My use of OSSEC is over the course of one week without any additional addons. Not adding addons is an adequate test of the product itself to understand the inconveniences that lead to the need for addons and in which areas. Immediately, the need for visualizing data was apparent. OSSEC alerts to a text file and with no inherit Web UI, GUI, or any other form of visualization, it was lack luster. The design is simple and less resource intensive, but the major drawback is update interval and manual intervention required. With a visualization component such as Kibana, it brings the capability to visualize data to enhance the ability of the perception of it since humans naturally understand pictures better than words. It also gives the analysts correlation and most importantly to me, automatic updates. Without any visualizer, I had to manually close the document, reopen, and scroll all the way down since alerts were appended to a text document. This may seem small, but it is an old and inefficient method in this decade. To OSSEC’s credit, with so many visualizers available like Kibana, Grafana, Power Bi, Tableau, etc. and while being understaffed, it is logical to leave it up to the engineer to integrate with those that are available, popular, and specialized, rather than building a meager one in comparison that people would switch over for anywhere.

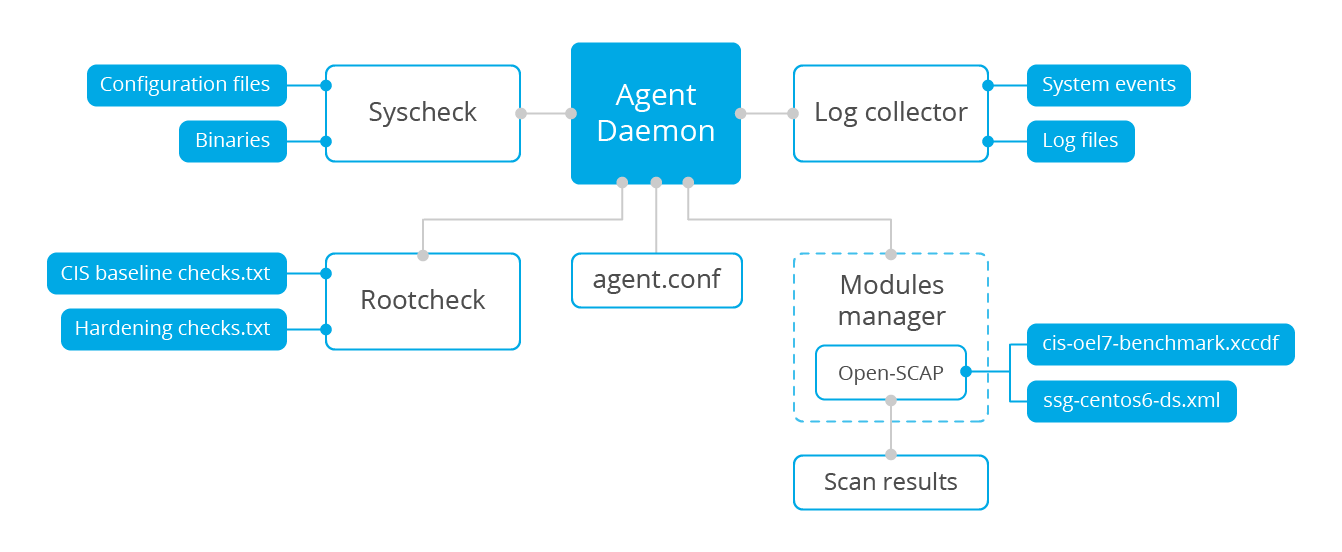
**Wazuh**[[3]](#endnote-3)

Introduction

Wazuh is a fork of OSSEC because of its lack of upgrades. Wazuh goal was to build a product that is more comprehensive, reliable, and scalable. It promotes clusting, automation, AES encryption, anti-flooding, multi-threading, RESTful API, improved log analysis engine, updated rules, cloud integration, compliance support, diverse vendor integration (e.g. ELK, VirusTotal, CIS-CAT, and more), CVE vulnerability databases and migration from OSSEC.[[4]](#endnote-4) In addition to integrating with OSSEC, it has combined with OpenSCAP-used to detect system configurations and vulnerabilities-and ELK-a suite that collects, parse, index, query, and visualize data.

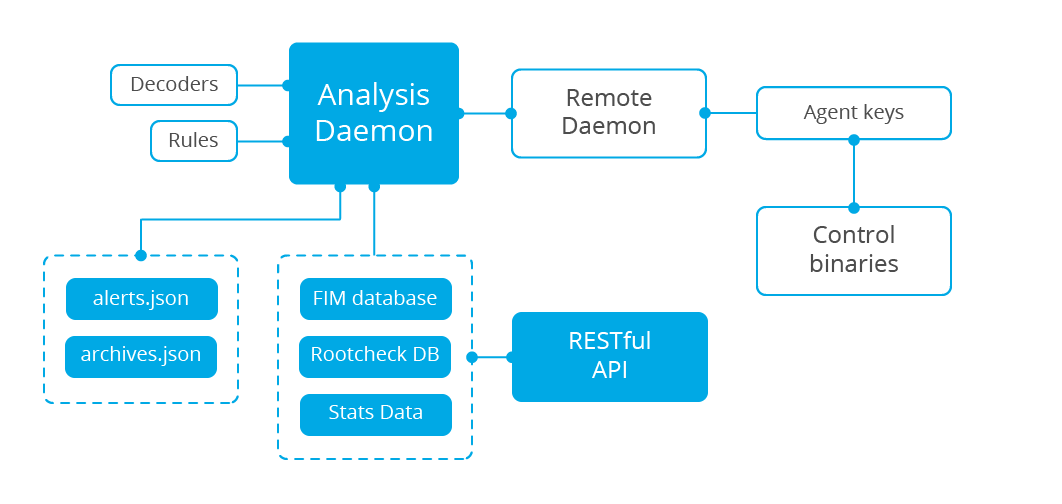
The various components are about the same as OSSEC, consisting of an agent and server. The agent can run on Windows, Linux, Solaris, BSD, and Mac. The data forwarded to the Wazuh server is secured through pre-shared keys. *Figure 2* below displays agent tasks. In summary, it’s a multi-functioned tool that detects root kits, collects logs, file integrity monitoring, scans for vulnerabilities, and forwarding the data to the server. Also, agentless devices are supported through there own means of forwarded logged data (i.e. syslog).

Figure 2 - Agent Functions[[5]](#endnote-5)



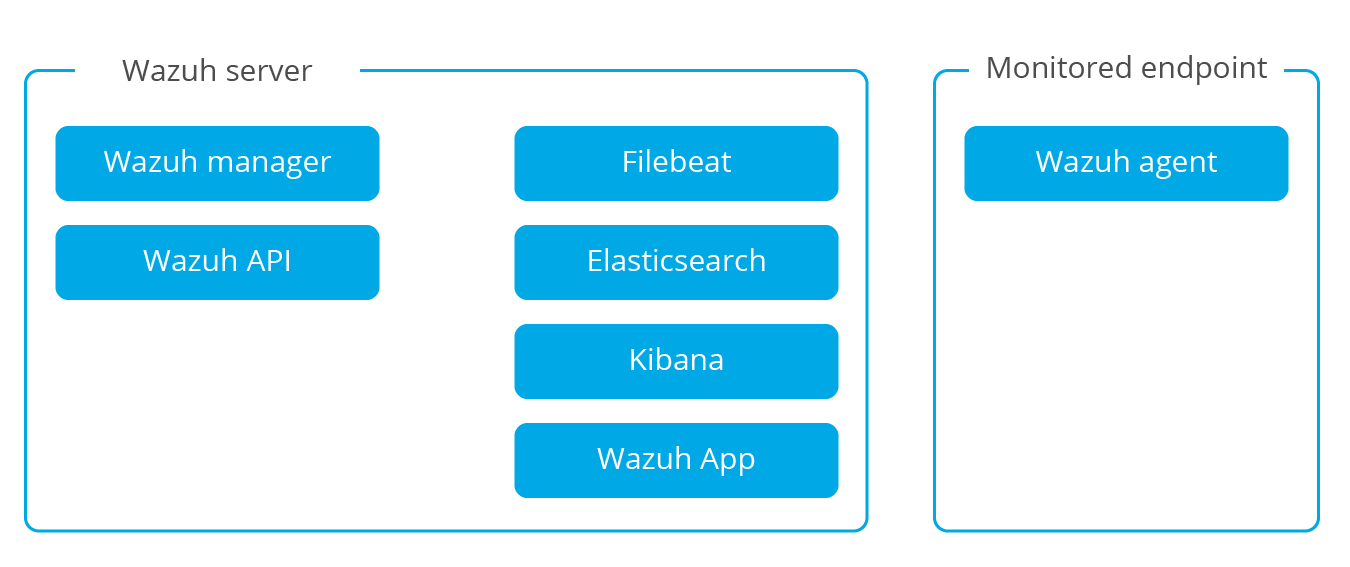
The server receives the data from the agents and alerts accordingly. The server also monitors itself. Such as the server in OSSEC, it includes agent management, alerts, and analysis, but Wazuh uniquely includes a modern RESTful API throughout its components. *Figure 3* below portrays a server function. Wazuh utilizes ports UDP/1514 for data from agents, UDP/1515 for agent registration, TCP/1516 for Wazuh cluster communication and TCP/55000 for API HTTP requests. These are the defaults, it can also vary between UDP & TCP, or use unsecured UDP/514 syslog for example. Encryption available is 192-bit Blowfish or 256-bit AES. Hashing is MD5 and SHA1.

Figure 3 - Wazuh Server Functionsv



In terms of architecture, our deployment will follow the single-node Wazuh server deployment. There are two available depending on the number of agents in the environment. The single-node is recommended for less than 50 agents and includes Wazuh server and ELK stack on the same host. The multi-node option is meant to be scalable for larger environments that requires a separation of the Wazuh server and ELK cluster on different hosts. We will only focus on the single-node Wazuh deployment, as displayed below in Figure 4.

Figure 4 - Wazuh Single-node Architecture[[6]](#endnote-6)



Installation

Installation is done in stages, Wazuh Server > ELK stack > Agents. I will be installation Wazuh version 3.11 on Ubuntu app running natively on Windows provided by the Windows store. The guide will be the documentation found [here](https://documentation.wazuh.com/3.11/installation-guide/index.html). The process is detailed below in *Table 2*.

Table 2 - Installing Wazuh

|  |  |  |
| --- | --- | --- |
| Installing Wazuh 3.11 on Ubuntu 18.04 WSL | | |
| Step | Description | Command(s) |
| 1 | Add Wazuh Repository | # apt-get update  # apt-get install curl apt-transport-https lsb-release gnupg2 -y  # curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | apt-key add -  # echo "deb https://packages.wazuh.com/3.x/apt/ stable main" | tee -a /etc/apt/sources.list.d/wazuh.list  # apt-get update && apt-get full-upgrade -y |
| 2 | Install Wazuh API | # curl -sL https://deb.nodesource.com/setup\_8.x | bash -  # apt-get install nodejs wazuh-api -y  # service wazuh-api status |
| 3 | Installing FileBeat | # apt-get install curl apt-transport-https -y  # curl -s https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -  # echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | tee /etc/apt/sources.list.d/elastic-7.x.list  # apt-get update  # apt-get install filebeat=7.5.1 -y  # curl -so /etc/filebeat/filebeat.yml <https://raw.githubusercontent.com/> wazuh/wazuh/v3.11.1/ extensions/filebeat/7.x/filebeat.yml  # curl -so /etc/filebeat/wazuh-template.json <https://raw.githubusercontent.com/> wazuh/wazuh/v3.11.1/extensions/elasticsearch/ 7.x/wazuh-template.json  # curl -s https://packages.wazuh.com/3.x/filebeat/wazuh-filebeat-0.1.tar.gz | sudo tar -xvz -C /usr/share/filebeat/module  # vi /etc/filebeat/filebeat.yml  output.elasticsearch.hosts: ['http://127.0.0.1:9200']  # update-rc.d filebeat defaults 95 10  # service filebeat start |
| 4 | Install Elasticsearch | # apt-get install curl apt-transport-https -y  # curl -s https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -  # echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | tee /etc/apt/sources.list.d/elastic-7.x.list  # apt-get update  # apt-get install elasticsearch=7.5.1 -y  # vi /etc/elasticsearch/elasticsearch.yml  node.name: wazuh  cluster.initial\_master\_nodes: ["wazuh"]  # update-rc.d elasticsearch defaults 95 10  # service elasticsearch start  # filebeat setup --index-management -E setup.template.json.enabled=false  # curl http://127.0.0.1:9200 |
| 5 | Install Kibana | # apt-get install kibana=7.5.1 -y  # sudo -u kibana /usr/share/kibana/bin/kibana-plugin install https://packages.wazuh.com/  wazuhapp/wazuhapp-3.11.1\_7.5.1.zip  # vi /etc/kibana/kibana.yml  elasticsearch.hosts: ["http://127.0.0.1: 9200"]  # update-rc.d kibana defaults 95 10  # service kibana start |
| 6 | Installing Windows 10 Agent | Download the [package](https://packages.wazuh.com/3.x/windows/wazuh-agent-3.11.1-1.msi) and follow the wizard. Be sure to keep the MSI as it is the only method of uninstalling it.  On the Server:  /var/ossec/bin/manage\_agents  Add agent  Extract key Copy key  On the Agent:  C:\Program Files (x86)\ossec-agent\ossec-agent.exe  Manager IP: 127.0.0.1  Authentication Key: <Copied string from the server>  Save  Manage > Start  View > View Logs // verify connection was successful |
|  | Uninstall everything | Server:  apt-get remove --purge wazuh-manager wazuh-api filebeat elasticsearch kibana -y  Windows Agent (as Administrator):  msiexec.exe /x wazuh-agent-3.11.1-1.msi /qn  PowerShell (as Administrator):  Remove-Item -path “C:\Program Files (x86)\ossec-agent” -force |

Issues I ran into were related to the WSL usage. If I chose to bind elasticsearch to my IP address, it will go through the new bootstrap checks for a properly deployed elasticsearch instance. This threw an error stating the following error within /var/log/elasticsearch/elasticsearch.log:

java.nio.file.NoSuchFileException: /proc/sys/vm/max\_map\_count

This error states that the associated file path is missing, and this is because WSL is not a fully blown instance, so there are some missing pieces. Windows has partially fixed the /proc directory by filling it with dummy values so that errors such as this would not occur, however, they didn’t reach this section. Luckily, I am only running this system locally, so binding it to my loopback interface (elasticsearch default act) is not an issue. Instead of configuring the /etc/elasticsearch/elasticsearch.yml file’s “network.host: <elasticsearch\_ip>” attribute, leave it blank. The other issue I got was the following after starting the service:

OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a future release.

Unlike the first message, this is a warning message and not an error; so, it elasticsearch will run. The other issue is that Wazuh did not include commands when setting up the server to have the services run on startup. Minor issue, but for clarity sake, is a basic inclusion. Another strange thing I’ve never seen before is on a Windows agent, there is no *uninstall.exe*, but they use the *.msi* packet installer. This requires you to keep the executable and be able to recall the command to do so. It seems economical, but I would have preferred the nominal *uninstall.exe* wizard.

Samhain[[7]](#endnote-7) [Discontinued]

A German-based HIDs that features multi-platform, PCI compliance, centralized management, file, host & self-integrity monitoring, logging, multi-vendor integration, and active response. It can be single-node or multi-node with a client/server architecture.

Installation is done by source and is detailed in Table 3.

Table 3 – Installing Samhain

|  |  |  |
| --- | --- | --- |
| Installing Samhain version 4.4.0 on Ubuntu 18.04 WSL | | |
| Step | Description | Command(s) |
| 1 | Dependencies | $ sudo apt update && sudo apt full-upgrade -y  $ sudo apt install zlib1g-dev build-essential libpcre3-dev libacl1-dev libattr1-dev gpg -y |
| 2 | Pull, extract and compile | $ Wget http://la-samhna.de/samhain/samhain-current.tar.gz  $ tar -zxvf samhain-current.tar.gz  $ gpg --keyserver pgp.mit.edu --recv-key 0F571F6C  $ gpg --verify samhain-4.4.0.tar.gz.asc samhain-4.4.0.tar.gz  $ tar -zxvf samhain04.4.0.tar.gz  $ cd samhain-4.4.0  $ ./configure –enable-network=server  $ make  $ make install |
| 3 | Customize |  |
| 4 | Set a baseline |  |
| 5 | Start |  |

I decided to scrap this tool for two reasons. The first being the installation was rough in the beginning. I only had two errors, but I wasn’t sure how to fix the second. When I attempted to *make install* the first time, I received the error,

make: \*\*\* No rule to make target 'scripts/samhainadmin.pl', needed by 'install-program'. Stop.

This prevented the installation. I went into the directory (~/Samhain-4.4.0/scripts/) and only saw “samhainadmin-gpg.pl”. Unsure of what to do, I made a copy of this file with the name “samhainadmin.pl” and tried it again and it continued as if the issue was resolved. With the first error gone, the installation began. The next error was the following,

trustfile: group member syslog not found in trusted users --> ERROR

I was unaware of what this meant at a glance so I could only return back to the *Makefile* and view the code. However, I started to think, “how would I get this on a Windows machine?” Turns out, I would have to install Cygwin, which I prefer not to do. With the error in front of me and this idea in mind, I decided to discontinue this project. I did not dive too deep into the documentation and with experience only amounting to part way through the installation process, my knowledge on Samhain’s capabilities are lacking, therefore, an invalid opinion.

AIDE[[8]](#endnote-8)

The Advanced Intrusion Detection Environment (AIDE) is the HIDS covered. It features a HIDS core functionality of being a file and directory integrity checker through the use of rules and hash checking to verify integrity. It may be deployed standalone or client/server. The only downside is that it does not support Windows natively. However, I’ve seen it recommended in Linux environments and this writing is to compare and contrast the different solutions available for implementation, so it perfectly fits the bill. Another drawback is it does not support large files (>2GB) by default but can be configured when building from source. Unlike the other products, there is no clear documentation as reference outside of the help commands. Installation on WSL is as follows in *Table 4.* Documentation is found [here](https://aide.github.io/doc/).

Table 4 - Installing AIDE

|  |  |  |
| --- | --- | --- |
| Installing AIDE version 0.16.2 on Ubuntu 18.04 WSL | | |
| Step | Description | Command(s) |
| 1 | Dependencies | $ sudo apt update && sudo apt full-upgrade -y  $ sudo apt install build-essential gcc flex bison libmhash-dev libpcre3-dev zlib1g-dev -y |
| 2 | Installation | $ sudo apt install aide -y  Mail config – Internet site [default]  System mail name – [default] |
|  |  |  |

As one would expect, with the number of steps, there were no problems.

Bibliography (Chicacgo Style)

1. https://www.ossec.net/ [↑](#endnote-ref-1)
2. https://www.ossec.net/docs/manual/non-technical-overview.html [↑](#endnote-ref-2)
3. https://wazuh.com/ [↑](#endnote-ref-3)
4. https://wazuh.com/migrating-from-ossec/ [↑](#endnote-ref-4)
5. https://documentation.wazuh.com/3.11/getting-started/components.html [↑](#endnote-ref-5)
6. https://documentation.wazuh.com/3.11/getting-started/architecture.html [↑](#endnote-ref-6)
7. https://la-samhna.de/samhain/ [↑](#endnote-ref-7)
8. https://aide.github.io/ [↑](#endnote-ref-8)