BASEC

An Interactive Guide of Basic Security

SSH[ghost] – Coder / Author

The University

January 2015 – June 2016

Abstract

BASEC ensures security by not only following the failing repetition of only advising average computer how to secure their systems and/or networks but also secures their machines for them. The goal is to create a multiple script system that can educate the user on basic security measures that are easy to follow, to secure their systems for them, and display how the system was secured so they may adapt the knowledge to future system set-ups. The primary script is an easy-to-follow bash script that displays the guide information, executes the secondary scripts, and gives the user control over how the flip through the guide. It's like a PDF in a terminal that can execute scripts. If they choose to execute the secondary scripts along specific areas of the primary script, they will find an easy-to-follow script that checks their security and, if prompted, make necessary adjustments. For example, the Gastly script checks for open ports, reports the statuses, and, if prompted, will change the open ports to hidden via iptable commands. **This is solely targeted to Ubuntu/Debian users**, but most of the commands can work on other *nix systems. Windows users will be provided a PDF that isn't interactive but still gives the same information (not innovative). Bash is too new to Windows to have tested if BASEC would function properly.

Keywords

BASEC, security, computer, system, network, script, educate, guide, port, vulnerability, exploit, iptable, Ubuntu/Debian, *nix, Windows, PDF, Gastly, Metapod, Autosnort, PerSec, NetSec, SysSec.

Background Information and Prior Art

BASEC is an interactive guide for teaching basic security. The intended audience was meant to be for every user type, but programming on Windows is more complicated that scripting for *nix systems that are already fairly open, filled with documentation, and with plenty of technical forums. Over time, it has evolved more towards a specific audience of users new to Linux and security, but the guide does still inform about Windows security too. BASEC was inspired by the lack of in-depth articles that attempt to, and fail spectacularly, to teach basic elements of security to readers and my former SIP that has been modified to be the Gastly script (Pagliery, 2014). While Gastly is one of the less important scripts and the final script, the focus on it through the Innovation Brief has to do with the script's previous status as my original SIP and one of three scripts. I created a second SIP and incorporated parts of the first that related to defensive security.

Gastly was inspired by Gibson Research Corporation's ShieldsUP!! using bash and *nix terminal commands to view which ports are open and then change the iptable rules to hide said ports. Gastly is the evolutionary step of ShieldsUP!! on a local machine to adjust the security of network ports via automation vs just reporting them. GRC's ShieldsUP!! is a browser-based utility to test a firewall for vulnerabilities with several options with the 'All Service Ports' option as the most popular, which scans the first 1,056 ports of a firewall (Ho, 2010, para. 1). What makes it stand out is that it is browser-based, thus doesn't require a download and installation, is fairly quick unlike commonly used penetration testing network scanners, and provides detail on both results of the scan and advice on how to make a firewall secure if the results returned said firewall as failing the scan test (para. 2). Steve Gibson (2003) explains the need for ShieldsUP!! to test firewall vulnerabilities for Windows systems is due to Microsoft's infrastructure of NetBIOS and NetBEUI which allows computers to connect with servers as well as any other computers to connect to one (para. 5). Actually, Gibson cleverly demonstrates the truth behind this using ShieldsUP!! since it connects to a users' computer in order to scan for ports in use. The vulnerability of a Windows machine is only increased with the use of TCP/IP File and Print Sharing (para. 12) or broadcasting a Windows machine as a local media server, although the latter requires access to the network to penetrate said Windows machine. Thus, the need for vulnerability scanning software is high.

Network Mapper, hereby mentioned as Nmap, is another utility for security auditing, but has many more features than ShieldsUP!!, is open source, and is a locally-run program ("Introduction," n.d., para. 1). As part of that long list of additional features, Nmap can be run on various Windows, UNIX, and Linux operating systems whereas ShieldsUP!! focuses on Windows. It is extremely flexible with command types to do a quite a lot more than just test the first 1,056 firewall ports but also various packet scans, OS detection, version number, ping sweeps, service uptime, and more. In fact, several lightweight penetration testing Linux distros that are capable of running off of a Raspberry Pi on an 8GB SD card can use Nmap such as the Raspberry Pwn ("Raspberry Pwn," n.d., para. 2). Nmap can even be used as a vulnerability scanner, but would require the knowledge and experience of reading network scan results to understand where vulnerabilities exist ("Nmap from Beginner," n.d., para. 1), which is how Nessus is popular since it's a beginner-friendly vulnerability scanner. Given the versatile nature of Nmap and the myriad features in currently offers, it's no wonder that it's the most popular network scanner (Burns, et al, 2007, p. 1084, para. 1) or that it's been featured frequently it films for its role with "hacking" such as The Matrix Reloaded, Live Free or Die Hard, The Girl with the Dragon Tattoo, and The Bourne Ultimatum ("Introduction," n.d., para. 2). However, Nmap is limited to its own purpose of network scanning and requires additional tools for actually adjusting ports, thus the purpose of Gastly to utilize both Nmap and iptables in a single and easy-to-use bash script.

In a similar fashion to ShieldsUP!!, books that educate users of either moderate or advanced security never secure an individual's system. The purpose of those books aren't to do so, but securing a system for a user and then allowing them to experiment from there ensures some level of security for the users until they become experienced enough to handle their own security. Maybe the author chooses to never learn security but leaves their system at the default security settings of the BASEC scripts. Many people don't bother to learn security or maintain up-to-date knowledge on the subject to keep themselves secure (Ferrara, 2012). While this is a hindrance to have employees not understand basic security measures, it's also up to the IT department to leave as little risk for such employees to cause damage as possible. We in the security field should make it as difficult as possible for new, ignorant, or lazy users to unintentionally cause harm due to a lack of understanding of their actions or lack thereof.

Metapod was a system hardening script based off of Cyberpunk's own guide. However, Metapod has been broken down from one script ten to use each separated function per each relating page instead of trying to run all the system hardening functions at once at the end of both the NetSec and SysSec chapters. Metapod and the other nine scripts automate the system hardening process unlike CyberPunk's guide, simplifies it for new users to either Linux or security, and keeps only the elements relevant to a normal system ignoring the server aspects ("System Hardening Guide," n.d.). In addition to the guide, some modifications have been made to include additional tools throughout BASEC in case if they weren't installed beforehand and to allow the use of Cron jobs. While the user should lock down who has access to editing crontab files, Cron is a powerful tool for automating numerous services, especially those relating to security like running Bleachbit nightly to clean the system or updating the system without user input (cogNiTioN, 1999).

BASEC isn't just to solve the issue of teaching new users basic forms of security without overwhelming them or to just secure their systems. The scripts were coded separately to be used and tweaked without the need of BASEC, yet still with the ease of reading as a new or amateur user. The ways in which the BASEC script bundle can be combined with other scripts is as limitless as there are scripts in existence. For example, AutoSnort can be used in parallel to Gastly and Metapod to install Snort IDS without needing much user knowledge or experience. The time to make security easy for everyone is now.

Project Description and Innovation Claim

BASEC is to be a Linux-focused, multi-script system with twelve bash scripts for those that are new to network security and/or Linux that want to immediately secure themselves as much as they want to learn about security. No one has created a guide that activates scripts to easily secure a system nor has anyone repurposed Nmap for securing firewall ports by probing the firewall and then cloaking the open holes. The BASEC script runs myriad functions to display sections of information like chapter sections of a textbook, runs a control bar at the bottom to jump anywhere in the guide, and executes other scripts and code where applicable per respective section. The Gastly script runs the basic port scan against the machine running the script, parses out the information in three Nmap files, uses cat and grep to move the open port numbers into another file, retrieves the port numbers from said file, and adds the DROP command to those ports for incoming traffic to hide them from attackers. The intention is to make hiding open firewall ports easy to detect and manage, which hasn’t been done yet on a purely home user level. The Metapod script, and other system hardening scripts, runs numerous commands from disabling unused system software, removing unneeded software, creating a partition for and securing /tmp, locking down SSH access and use, creating port knocking to still use SSH, adding IPSEC, disabling wireless services for their inherent vulnerabilities over wired networks, etc. Like Gastly, the system hardening scripts require users input to actually make changes while warning of the effects of such in case a mistake is made or in case they want to use an affected service. The function to disable wireless services from other, usually unused tools was for this very reason. There no longer exists a system hardening script for Debian-based Linux users and there never was one that was easy to read in output or code (“Bastille

Linux," n.d.). If any one script isn't innovative enough, the combination of all three ensuring basic security and teachings of basic security are ensured, thus innovative in this claim.

Usage Scenario

As BASEC exists currently, there may not be much use for BASEC in the future as just the guide. It makes sense to use it in the beginning when the user is new to cyber security and/or Linux. However, it makes more sense that as one's own experience and knowledge progresses so too does their understanding of scripts. The script version of BASEC may not be needed after the first few studies, especially when there's also the PDF version that's not interactive that can be used for reference. The other scripts were coded as separate scripts for this very reason and so the user can make use of the secondary scripts without the need of BASEC. It's perfectly logical to keep the secondary scripts to tweak or use when needed as they serve a functional purpose, but the BASEC script (not the guide itself but the script version) itself does not once the user learns how *nix commands and scripts function. They can be adapted and tweaked, but the BASEC script only serves the purpose of giving the information of the guide while automating the secondary scripts for the user.

Here's how BASEC functions. Upon running the script in the terminal (as will be advised on the site) with root permissions, the user will be greeted with a Pokemon ASCII art that has downloaded and deleted in the background. Adding ASCII art to a bash script is difficult when certain characters are used that would break the display function and the scripts should have some imagery to display on executing. They will then see the title page with the subtitle and my coding username. At the bottom of the page, and every page, is a control bar that allows the user to move to the next page, return to the previous, jump to the table of contents page, jump to a specific page number, or exit the script. Granted, there are no real pages but functions instead. BASEC was coded with the attempt to mirror an ebook inside a terminal, thus the literary names are borrowed. The next page will include my name and information, then another page for my former partner's information (our SIP's will prove complementary, thus I maintain his name on the project after leaving), author's note and warnings of changes, table of contents, and then the guidebook chapters.

Most of the functions for the complementary scripts have previously been written. However, there are additional details of all three scripts. They are color-coded for even easier understanding for the user, even if the colors may not be the norm. Cyan displays general output, green and red are used for yes and no, magenta is used for warnings and additional information, yellow is used for terminal feedback given by the system upon entering certain commands, and white is used for user input. The user input, excluding page control of BASEC, is boiled down toward entering 'y' or 'n' for continuing on a specific task once brief of its purpose. Some functions do not add this since they aren't necessary like Metapod's functions to confirm the system is up-to-date or entering the IPSEC information. Once the complementary scripts finish, they display what would be the next page number before allowing the user to exit or return to BASEC. Granted, the need to return to the specific page vs listing the information seemed to be a waste of time for what would be an easy task for the user to complete otherwise. The same goes for the use of '+' and '-' to navigate pages in place of arrow keys.

Apart from the aforementioned, the scripts were coded to be easy to understand. Instead of laying out the scripts in a seamless display of commands, functions were used so that the user's, upon editing the script, would know what code to search for where. Each first step of its type has also been documented so that the user can easily follow the code. Even after a user gets beyond the need for BASEC, there's no reason why the code can't be easy for them to read and tweak as they see fit. BASEC also includes functions where scripts would otherwise be used to perform specific functions such as downloading Bleachbit. No script was given for this function sine it's only three commands long and can be done just as easily from a browser GUI by any type of user. This project was coded with simplicity in mind.

Evaluation Criteria

Through constant testing and advisement of professional, experienced, and amateur coders alike will the necessity, direction, and capabilities be accessed. The following criteria will evaluate the effectiveness and simplicity of the scripts for security and / or Linux beginners:

- **Simplicity –** Can the user understand how to run the scripts and read the results? Will the user know which ports to hide, which software to disable, which functions they may want to skip, or how to run the scripts and commands without the primary script?
- **Effectiveness –** Are the scans as thorough as necessary? Does the system become hardened while maintaining functionality?
- **Time/Resource Usage –** Are the system hardening functions quick or light enough that it won't freeze or break the system (like the /tmp function)?
- **Practicality –** Are the scripts unique enough that they are practical enough to use?
- **Functional –** Are the scripts still usable or are tweaks needed to make them up-to-date (like AutoSnort's Debian installation script)?
- **Diversity –** Are the scripts written well enough that the user can add their own commands and functions to the scripts?
- **Organization –** Are the scripts packed together with the results in a way that allows their use without the user otherwise knowing a temporary results page existed before self-deleting?
- **Stability –** Are the scripts constantly tested and updated of any and all types of errors?
- **Uniqueness –** Are the scripts beneficial to the user that they may be adapted?
- **Cross-platform –** Are the scripts capable of being run on Windows and UNIX systems?
- **Additions –** Will adding more scripts be useful and allow for additional effectiveness and practicality?

Project Logic Model

Goal – Create a user-friendly, interactive guide that teaches security, allows navigation, hardens the system, and adds additional security layers.

Objective 1 – Create a guide that teaches users the basics of PerSec, SysSec, and NetSec.

Activity 1 – Research tools and methods advised from numerous sources

Activity 2 – Write own advice on top for what advice has worked

Activity 3 – Provide at least 2 tools per each page

Objective 2 – Create a script that scans all the IP addresses of the localhost and hides open ports

Activity 1 – Download and install Nmap

Activity 2 – Create command to run Nmap with ICMP pings

Activity 3 – Create command to store results in all Nmap results files

submenu to list results and time scans take to complete

Activity 4 – Use cat and grep to parse out open port numbers from the Nmap files

Activity 5 – Use iptable commands to DROP incoming packets to those ports, save the new firewall rules, and reset the firewall

Activity 6 – Create a command to warn to reboot system for changes to take effect

Activity 7 – Create a command to display next page # of BASEC and to return

Objective 3 – Create a script that hardens the system

Activity 1 – Update system and add unattended upgrades

Activity 2 – Install widely used security tools

Activity 3 – Use commands to secure terminal, permissions, and memory

Activity 4 – Use commands to secure Cron and SSH access

Activity 5 – Create port knocking procedure to use SSH

Activity 6 – Use commands to create /tmp partition, move /tmp, and link /var/tmp

Activity 7 – Use commands to disable / remove services and wireless

Activity 8 – Add IPSEC rules and use commands to secure firewall

Objective 4 – Create script to house guide and run other scripts

Activity 1 – Add guide to script

Activity 2 – Separate sections to own functions

Activity 3 – Use commands not scripted between sections

Activity 4 – Add pages between sections to run scripts

Activity 5 – Add control bar to navigate between sections

Prototype Implementation

```
                 |`._         |\
                 `   `.  .    | `.    |`.
                  .    `.|`-. |   `-..'  \           _,.-'
                  '      `-. `.           \ /|   _,-'   /
              .--..'        `._`           ` |.-'      /
               \   |                                  /
            ,..'   '                                 /
            `.                                      /
            _`.---                                 /
        _,-'               `.                 ,-  /"-._
      ,"                   | `.             ,'|   `    `.
    .'                     |   `.         .'  |    .     `.
  ,'                       '   ()`.     ,'()  '    |       `.
'-.                    |`.  `.....-'    -----' _   |         .
 / ,   ________..'     '  `-._              _.'/   |         :
 ` '-"" _,.--"'         \   | `"+--......-+' //   j `"--.. , '
    `.'"    .'           `. |   |     |   / //    .       ` '
      `.   /               `'   |    j   /,.'     '
        \ /                  `-.|_   |_.-'       /\
         /                        `""          .'  \
        j                                          .
        |                                 _,        |
        |             ,^._            _.-"          '
        |          _.'    `'""`----`"'    `._      '
        j__     _,'                         `-.'-."`
          ',-.,' mh
```

BASEC and the complementary scripts were coded for Ubuntu Linux. While BASEC can be used on any system that uses bash, some of the scripts were designed for Debian/Ubuntu and the only version of AutoSnort that works, in regards to Debian and Ubuntu, is Ubuntu. Autosnort is not my script, but has been included due to its simplicity and power to install the popular intrusion detection system Snort. BASEC has been designed in such a way that the user only needs to run BASEC itself and the other scripts will be executed by BASEC when prompted in the specified areas respectively. For example, Gastly isn’t executed until after the section about firewalls and after the user selects to run Gastly in between the firewalls and NEXT CHAPTER.

This works the same as Autosnort where it runs the complementary scripts at specific sections of the Autosnort script.

It's advised to 1) run BASEC the complementary scripts in the home directory, 2) to run it on a cleanly installed machine, and 3) run it in a terminal that snaps to the left or right side of the screen. BASEC assumes the users truly are new to security thus having no modified security from the default installation. BASEC can also be used to automate securing a freshly installed Ubuntu instance, thus why the complementary scripts are their own scripts and not coded into BASEC. BASEC has also been coded to run the terminal in half-screen to do whatever else in the other half or to view the code side-by-side with the terminal running the scripts.

BASEC is executed by opening a terminal and entering the following without quotes:
"sudo ./BASEC.sh"

BASEC as a guide functions by teaching users new to security and/or Linux the basics of security. Pardon my wording, but the three parts of the guide are broken down into three "chapters" (instead of parts) and each page is a different section of the guide or part of an ongoing section. The three parts are PerSer, SysSec, and NetSec or personal, system, and network security. There's a lot of focus on personal security as it's the most important, isn't limited to just computer systems, and is often the least taught. The complementary scripts are what functions for the NetSec and SysSec chapters. BASEC was coded with those three chapters in their particular order due to making NetSec the least concerning. In case of network issues of conflicting set-up, the SysSec scripts won't function.

BASEC's navigational control bar functions by carrying out a command based on user input. Entering 'T' accesses the table of contents, entering 'N' jumps to the next security script page prior to running the script, entering '+' moves to the next page, entering '-' returns to the previous page, and entering 'Q' exits BASEC. Each page header and navigational control bar are color coded for the reader's easement. The commands previously mentioned in apostrophes are in cyan, the short description in magenta, the dividers between each function are in red, the 'Option:' output is in green (as well as page headers), and the input is in white (as well as the general text of the guide).

Pokedex is the alternative primary script to the BASEC without the guide. It acts as a menu for the complementary scripts if the user chooses not to run through the whole guide to run the complementary scripts, but doesn't want to type them out each time as well, especially since running them freely lacks any organization. Pokedex is the only script in the collection that doesn't use ASCII art, but that it due to its purpose of brevity as a menu script for the BASEC complementary scripts. The initial display is of three categories: SysSec, NetSec, and Extra. SysSec hosts the five system security scripts to be described below and NetSec hosts the five network security scripts to be described below. What Extra houses are the commands that were not made into their own scripts that were added to the intermission pages between each page such as installing bleachbit and AEScrypt, downloading and running Autosnort, and downloading pwd.sh, Netdata, and DNScrypt.

Each script also starts by retrieving an ASCII art photo created by Maija, clearing the screen of the wget function, displaying the ASCII art, and then deleting the file downloaded. I felt my scripts needed some sort of imagery when initially run and trying to display ASCII art in-script is a pain. The files could have been downloaded and added to a resource folder, but I felt this was cleaner for the small data used. There is a theme of Pokemon. One reason has to do with my coder ID. Another has to do with how many ASCII art files there are to work with that can fit in the terminal. The last and primary reason has to do with each Pokemon chosen is in some way relevant to the script displaying it. Gastly uses an image of Gastly because the script hides ports or make the user invisible to port scanners like a ghost. Metapod uses an image of Metapod because the script conducts system hardening like Metapod famous move of increasing defense with the attack 'harden.' Cloyster was used over Shellder for securing secure shell since the Shellder ASCII was bad. Each script has a semi-relevant image of a Pokemon.

The complementary scripts are color coded is a similar fashion. Script output is in cyan, extra info or warnings are in magenta, and errors are in red (as well as the 'n' output for defining 'no' of user input). If a command has any output, the output comes out yellow. It's the only output that isn't scripted by me since it's output of commands used. User input is still white and 'Option: is still green. Green also functions inversely to red with the option 'y' for 'yes.'

The complementary scripts are ran in the following order:

SysSec

1.	Metapod	System hardening
2.	Haunter	Adding user to Cron to automate scripts / commands
3.	Grimer	Disables / removes software not generally used
4.	Koffing	Disables webcam / mic
5.	Machamp	Moves /tmp into own partition

NetSec

6.	Muk	Acts like Grimer for network services
7.	Weezing	Disables Wi-Fi / Bluetooth
8.	Charizard	Sets up firewall
9.	Cloyster	SSH hardening
10.	Gastly	Filters open firewall ports

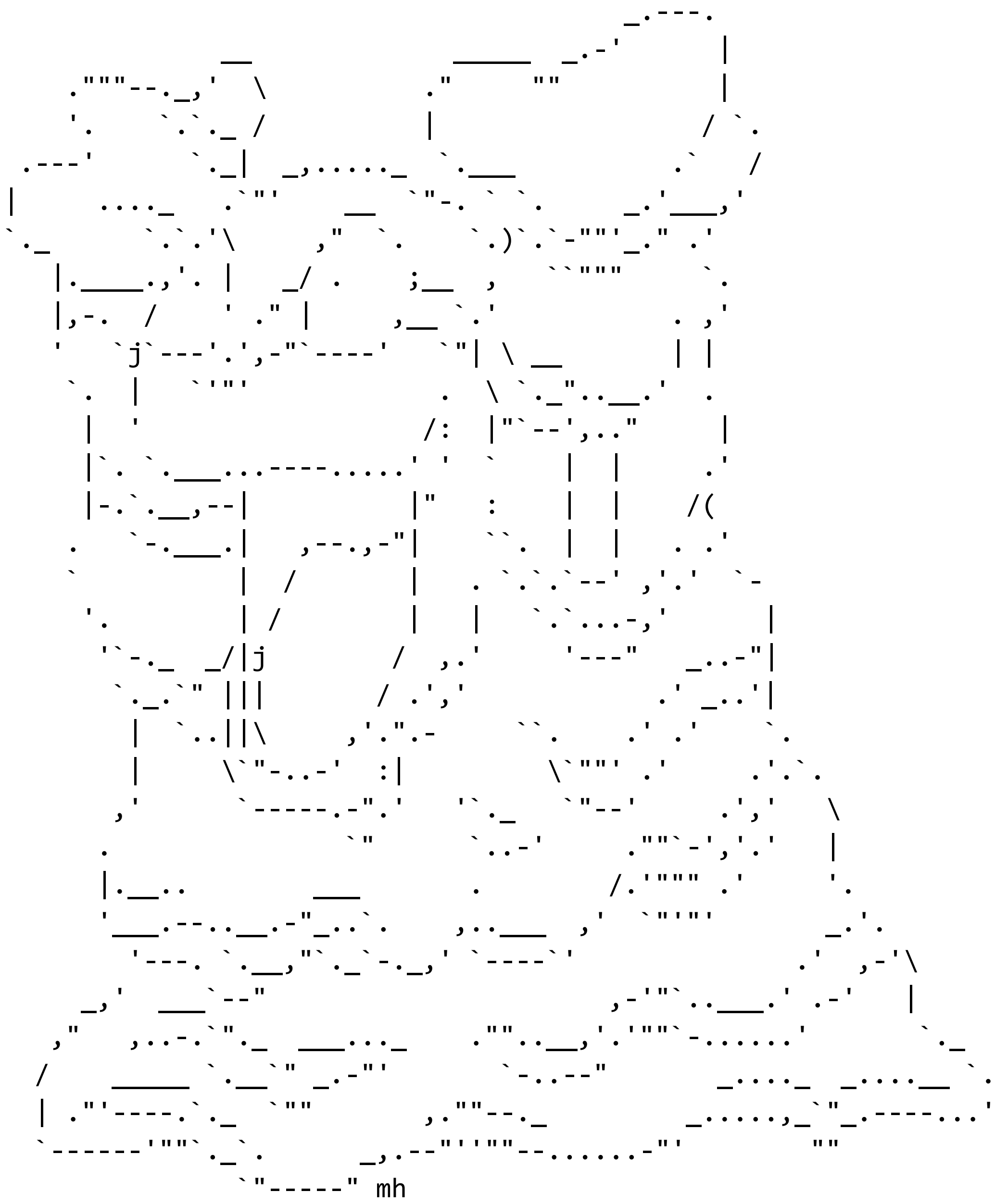
Autosnort is ran last and is NOT my script. In fact, BASEC will ask after the IDS chapter if the user wants to install Snort. If 'y' is entered, BASEC will download Autosnort, extract the data, move to the Ubuntu directory, and then ask for input again to run Autosnort. The reason for this is that Autosnort was coded to have no user input apart from entering their MySQL password and oink code.

```
                                   ,--..
                                  /     `.
                                 /|       `.
                                / |        |
                               /  j        |
                              /  |         `
                             '  ,'          \
                           ,'                L
                          /                  +
                        .:.                  `,
                     ,"`.  `.       ,..-._    +
                     |  |`.  L     '   _.'`.   .
                     j  `.,\ '    | ,.' |  +.  +
                    '`.    |,'    |" `""   / `, .
                   |   `""'/      `-.____.'    \|
                 ,'|     ,'                     Y
                /  |    /                      '|
               /   |  ,'                     ,' +
              /    \-'                      /    `
             /    /                       ,'      `
            .     ,`'-.                 ,'         L
             \   /     \               /            .
                /      `               \            |
              `/          _,            `          ,'
               |                         `       ,'
               |           "'             `.   ,'
               j         -"'               |`-'
              /                           /'/
             /           ,               / /
            /            '              j /
          .' ___                        '/
          |-'   `"`-.                  '/
          '          \                .'
        ,"            l          _,.-'
       ,---..         |L     _.-'
     ,'      `.      / |  ,-'
    /          `  _,'  ;-'
  ,'--.       ,-`|  ,-'
 /     L   _,'  _|-'
(       \-' _,-'
 `......^.-' mh
```

Metapod functions as a collection of commands that deal with general system hardening that isn’t large enough to be its own script (like Cloyster is a system hardening script specifically about SSH). It updates the system, adds unattended updates, installs necessary tools to use throughout the use of BASEC, limits su to user, secures shared memory, disables all access to cron or SSH (until specified in Haunter and Cloyster), etc. Haunter, Muk, Machamp, Charizard, and Cloyster used to all be under Metapod. It made the script too long for specific functions and meant each function could not be used after each appropriate page.

```
               -._                                  _.
                \ `-.._                          _,' |
                 \    `-._    _,.--------.._  _."    '
                  \       `--'              ``.     /
                   \                               j    __
__         __       \                              |.-"' /
 `.`-.`-.__`.`'"----"\                             |    /
    `.       `.       '       ._                       /
    `..        \              | `.               /|   /
      `.        `.            |   `._          .' |  /
        `.  .-----`           |      `.       /   ' '""''
          `. `.           .    ._      `_    /  ,'    .'
            `. `.          `._   `'""'"'     ""' ,  ,'
              `. `.         `.`.              ,-/ ,'       _..
                `. `.         \|,---..  ,--"./ / ,--------".  \
                  `._          `.     `/ , .`.',:           \  \
                    `._          `..".,./ ' _.' :            \  `.
                  ,-'" `-._              _."     .   |.-"`.   \  |
                 .         `-..........-'        |   `..   \   |_'
                 |           `".                 `.._   .  '  ,'
                 |         |   |                     `"'    .'
                 |   /\    |'  '
                 '  /  \   ||   .
                '   \  '   |'   ;
                 \  '  \   `...'
                  `""   `,' mh
```

Haunter functions as a change to Metapods’ function to block all access to Cron. Instead, Haunter has the user enter their username so all users but the entered username are blocked from Cron. This script was made due to the high value Cron offers all users of all types for its ability to automate anything based on time, date, month, or day of week. At 2300 every night, Cron runs my updates. At 2345, Cron runs bleachbit to clean my system. At 2400, Cron dismounts my mounted, encrypted partitions and shuts down. In BASEC, the user is taught about the value of Cron, how to use it, why we don’t edit /etc/crontab with an editor instead of crontab, and examples of what they can run when. Haunter ensures they can still automate security commands / scripts that aren’t already running as daemons upon boot.



Grimer functions be disabling and / or removing aspects not generally used that would otherwise be major vulnerabilities. For example, the root account is disabled and the user is taught the difference between the root account and their own access to su. It disables anacron and compilers upon input as the user may find use for them (I do), but they are unnecessary if not used. The next changes are disabling and removing the ability to automount removable media and disabling and removing at (it’s redundant when Cron exists). There is an additional function that is separated to be used by itself to disable and remove zeitgeist. However, if the user is using Ubuntu and not an Ubuntu-based distro or modified Ubuntu instance like Xubuntu, they are advised to skip this and view a link given to disable the service (disabling it on an actual Ubuntu system breaks Unity).

```
                               ,----.
                              '      |
                             /       '
                       __,..'         "-._        _
                  _.-""                   `-.   ," `".
         ,-._  _.'                           `"'      '
       ,'    `"                                       |
      .                                               .
       `.          _.--..               ____          '
       /         ,'    . `           ,"' .  `.         `.
      /         .         |         /         \          \
     /          `------...'        ._____      .          \
    .                                    `'"""'            \
    '                    ________                           .
   j           `.""/'"""`        '"""'"'--....,-            |
   |             `/.                      ,\ /              `.
   |                `-._               _.'  '                 `-.
   |                    `"-----------"'                         |
 ."                         ____                                |
|                      ,-""'    `".                            ,'
|                     .   .----.   `.                        ."
`.._                  |  '.____,'   |                        '
    |             ,". `.           ,' _                     /
    '            '   `._`.'._".__,' .' .                   /
     .            `'-._ `._     _.-'  _.'                 /
      `.               `.  `--'" _,.-'                    `
        .               ,'     ."                          '
         '        .-..-' _,.--._`"-..,-.                 ,'
        /         \    ,'       `-.    |           .-'"-"
        \          `-.'            `..'         _,'
         `.,.-"`._                           ,-'
                  `"-.                       |
                      \       ,..----.     _.'
                       `""---"        `..-" mh
```

Koffing functions by disabling webcam and camera. Attackers can use those to spy on unsuspecting users without their noticing since webcams can be enabled without turning on the light. The function to disable both of these has been separate from Grimer as users are likely to want to keep their webcam and mic enabled like Wi-Fi and Bluetooth.

```
                 __."`. .-.                     ,-..__
              ,-.  \  |-| |                ,-"+' ,"'   `.
              \  \  \_' `.'              .'  .|_.|_.,--'.
               \.'`"     `.               `-' `.   .  _,'.
                \_      `"""-.            ."--+\   '"   |
                | `""+..`..,'             `-._ |        |
               j     |                        '.       _/.
              /   ,' `.       _.----._          `"-. '     \
              |   |     |   ,'  ,.-"""`.           |  .     \
    __        |   '    /-._. ,'         `.         |   \     \
   (  `.     `.     .'     | /  _,.-----. \       j     .     \
    `. |.  __  `,          |j ,'\        `|"+---._|           ,
 .-"-|"' \"  |   ". '.     ||/d |_-""`.    /     ,'.           )
 `._. |  '.,.'     '  `  ,||_.-"       |  j     '   `         .
."'--:' .  )         `.  (     _.-+    |  |                   |
`-,..'  ` <_           `-.`..+"   '   ./,  ._          |      |
 `.__|  |  `-._       _.-"`. |   /  ,'j      `. `....' ____..'
   `-.,.'    \  `. ,'      ,-|_,'  /  |        `.___,-'   )
      `.     `.   Y        `-..__.',-'    __,.'           '
        `         '   ,--.     |  /            `+""       `.
         `.       ,--+   '  .-+-"   _,'   ,--  /     '.    |
           `-..   \     __,'            .'    /        `.  |
               `---)    |  ____,'      ,....-'           `,'
                  '                  ,' _,-----.         /
                    `.____,.....___.\ _..._______________/
                                   __\:+.`'O O  O O  O |
                               ,-"'  _,|:;|""""""""""""|
                             ,'   ,-'  `._/    _."  .`-|
                          .-"    '      \    .'      `.`.
                         :      .        \   |        / |
                          .      \.__   _,`-.|       /  |
                          `.      \  ""'     `.         `....
                            .     |            \              `.
                           .'   ,'              \               |
                  ,------'      `.               `-...._  '"-. '.
                 / ,'"'"`         |                  `--`._       `.
                 `".....---""--'                          \        .
                                                           |         `.
                                                          (   -..      .
                                                           `"""'  `....' mh
```

Machamp functions be backing up /tmp, creating a partition of 1GB, moving /tmp to the new partition, removing the backup, and mounting the new /tmp directory. Then it move /var/tmp, creates a link to /tmp, and copies the files of /var/tmp to /tmp. This is to secure temporary storage space so attackers cannot store anything malicious here during an attack attempt. Users are advised to NOT use this if on a system using FDE, to set this partition up upon future installations during the install process, and then run Machamp.

```
                             _,..-------.._
                         _,-'              `-._
                       ,"               .---.._`.
                      /----.       _._,.._""":"`"`.
                     /`--""     _.'.'  / .`-.'-"'"`\_
                    /""-._`----" .'  .' '   _.`     \"-._
                   .      `.___.'   / ,' . |   |  ._  . .``-._
                 _,' _.            .,' ,"| |   |   .`.|'.' ,--".
               ,'    ||`._..____.."' ,'--'.|   |    `.|'  .'"-._|
              /___...'|   :._     _,'      `'-. |     |  _,.._  |.
             |,--.._|j  ,'   `'""'         ||  `|     '.'_..._` | )
            ,'      /  /                   ||   |    /',"     ` '/|
         ,-'   .,-./  /      _,..----..___,':`_'    /.'        `.' ._
        /  __.._ `-\.'-----"'  _,...__      '"     /.""-..-"""-. |_  `._
   __..:--"_.--.`.  `---....-`"..__." `-...      _."'"''`---..__`._`""__\
.-'  `""''"       `..--""-.   _.--.._   _.-""-_,'              _.._`-._  '_
|""-.    __.-----._ '....`._-" _,-"'`-..._,..`__..."""-.__  _.'_.._`-..:'"-)
 `""`..-"_...----..___)     `""              `...------._"",-'     `..-'`""
      `"                                                  `" mh
```

Muk functions by disabling and removing generally unused networking services that may be exploited like network file sharing (NFS), crash reporting (apport / whoopsie), network printing (CUPS), and more. Telnet and snmp are especially disabled and removed. This has been separated from Grimer due to the network aspect and since it doesn’t belong in the SysSec chapter.

```
                  __....____,'  `-.
         ,""-..-'"          "-    |       ..      _.._
         |        _, .,           '._    /  `'"""'    |
        .'         _____             `.,'     ____     `.
     _," _.'      \  |  '"--..        '       \__ `"-.   `.
    | ,'"|/        `-.______,'      `     ' |\_  `'""'  .  \
    .'---'      _____             . '   `   |,'"""-._  ' \  .
   /   __..--""|___/ "-.._/|         ,'       ___    `. \ \ '_
  . .'...-----'""----.._.' |-.      |        | ,.`".   \ `'   |
  | `"                  "-.'-'      `.        \`._`.\   |.  .-'
."        _..._                 .              `._  ,  `.'  |
 `.    ,"'_....`".               |                ""       .
  |    | |     | |             -'   `,                    /
  `,.  `._`---'_,'  ,-.    '        ' `-.             _   |
  ,\ `._  |___|  _,'  |  `    \        . /-.__     _.' `-"
  \ `.._`-._ _.-'_,.--'        |        `.    |  ,'
   `-"\ `-. ' ,'_         `-..'       .-'    /  /
       `._.---._ `"----.        .   ,'.   _.'  /
         `.     `'-.._/       -" ,-" `.`-'      `.
           `.,       .-"    _    |     | .     ) |
             '._  ,"`----""`.    |     ' `'       .
                ""           `--'       \`      ,"'
                                         `-._,-' mh
```

Weezing functions by disabling Wi-Fi and Bluetooth. Attackers can intercept the publicly broadcasted radio waves, crack the packets, and learn the Wi-Fi password to set up a MiTM attack to fully monitor the users network traffic. The function to disable both of these has been separated from Muk as users are likely to want to keep Wi-Fi and Bluetooth enabled like their webcam and mic.

```
                 ."-,.__
                 `.     `.  ,
              .--'  .._,'"-' `.
             .    .'         `'
             `.   /          ,'
               `  '--.   ,-"'
                `"`   |  \
                   -. \, |
                    `--Y.'      ___.
                         \     L._, \
               _.,        `.   <  <\                _
             ,' '           `, `.   | \            ( `
          ../, `.            `  |    .\`.           \ \_
         ,' ,..  .           _.,'    ||\l            )  '".
        , ,'   \           ,'.-.`-._,'  |           .  _._`.
      ,' /      \ \        `' ' `--/   | \          / /   ..\
    .'  /        \ .         |\__ - _ ,'` `        / /     `.`.
    |  '          ..         `-...-"  |  `-'      / /        . `.
    | /           |L__           |    |          / /          `. `.
   , /            .   .          |    |         / /             ` `
  / /          ,. ,`._ `-_       |    |  _   ,-' /               ` \
 / .           \"`_/. `-_ \_,.  ,'    +-' `-'  _,        ..,-.    \`.
.  '         .-f    ,'   `    '.       \__.---'     _   .'   '     \ \
' /          `.'    l     .' /          \..      ,_|/   `.  ,'`     L`
|'      _.-""` `.    \ _,'  `            \ `.___`.'"`-.  , |   |    | \
||    ,'      `. `.   '       _,...._        `  |    `/ '  |   '     .|
||  ,'          `. ;.,.---' ,'       `.   `.. `-'  .-' /_ .'    ;_   ||
|| '              V      / /           `   | `   ,'   ,' '.    !  `. ||
||/            _,-------7 '              . |  `-'    l         /    `||
. |          ,' .-   ,' ||               | .-.        `.      .'     ||
 `'        ,'    `".'    |               |    `.        '. -.'       `'
          /      ,'      |               |,'    \-.._,.'/'
          .     /        .               .       \    .''
        .`.    |         `.             /         :_,'.'
          \ `...\   _     ,'-.        .'         /_.-'
           `-.__ `,  `'   .  _.>----''.  _  __  /
                .'        /"'          |  "'   '_
               /_|.-'\ ,".             '.'`__'-( \
                 / ,"'"\,'               `/  `-.|" mh
```

Charizard functions by setting up IPSEC rules in the system configuration and setting up the firewall. The latter includes allowing HTTP and SSH while blocking telnet and everything by default, adding a list of preconfigured firewall rules to ufw, blocking IPv6, blocking ping floods, blocking IP spoofing, and changing host information. I expect more trouble from this script than Gastly if the user already has attempted to set up firewall rules. Instead of simply flushing iptables and saving changes, the user is advised to keep in mind the changes they had previously made and ensure no conflicts are present on their next session. They are advised to flush iptables if that is the case.

```
                                                          _
                                                       ,"  '
                                                     ,'   /
                                                    /    /
                                                  ,'    /
                                                ,'     .
                                     _...      /       '
                                  ,'"'"".`.   |_..    /
                        ,-"'".   /       \\  ,'   `."'
                       ."""'. |,'         .`".    /.
                      /     | |           `...`".',___
                     /      | `.            _.-' /.___`".
                   .'       `-._`-.._____,-'    /     \ |
             ,..    \           `. . ,'.'      /       .|
           .'   `.   \   /        .||.'       '.`.._   ||--.
           \      `...`-'         ||||      .'  `-.._`.||_ |
       _,.--`"'-.."'              ||``. ,-""""`'"--._`"-._`-.
  _,-"'           `.              ' .\`'             `.   `. \......
,'                 '               || .|              |`.   \ \   .'
`..____________ ,'"`+-""'''       || ||              |  \  `. .-'
               |    .    _..-'    j | ||              |  |    ||
               |    |   .         |. j |              |_'|.   ||
               |    '    `-.._    || | |  .........      | |  ||
               |     `._          || ' '   \  __.-"  _,' '.. , '
               `._      `..       ' . `..   ""_,..-"'  .'  \/.'
                  ``.    /         \`.  ``-._`-...."_,'   .,'|`._
                     `..'           `.`._  `.`--"'      .,'\ :   `.
                      /     _.-   .    `-.:-._  ` '._   :   . `--..'
                    ,'  _,-' `.   :    __  `-.` `.._`. /,  ,','
                  .'  .'       `-. `-'"       \`.  `',   ','
                  `--' `--..     | `-._____   `._:---.| .'/
                            \    |     .'  "-.____`". |-"'
                             `.__|    /           `"'"
                                 |   /
                                 '._' mh
```

Cloyster functions as a change to Metapods’ function to block all access to SSH. Instead, Cloyster has the user enter their username so all users but the entered username are blocked from SSH. This script was made due to the high value SSH offers all users of all types for its ability to transfer files or commit any action remotely and securely from machine to the next. After the user enters their username, Cloyster adds iptable rules in addition to Charizard to better secure SSH by blocking incoming pings, locking out anyone for 24 hours for attempting to port scan, logging the attempt, and allowing the user 3 attempts to connect to SSH within 30 seconds. The next function establishes port knocking so as to hide the SSH daemon from an attacker and allow the user to communicate with SSH over any 3 chosen ports above 1024. The last function is to modify the SSH configuration files to be more secure and then restrict access to those files so only the user can make changes.

```
                           _
                         ."' `..._
                        '        `.
                      .'     ___..'
                _   ."       '  .__,-.,"", ,----.
     ,.-""''-.." :  :       `--'        ' :      :
    .'           :_,'                   `._`"--. ;
    :             _,.--'"'""`--._          `.  `"
   j            ,'              `-.      ,._.'  ,"".
   :          ,'                  ,-.   .   __  `..'
   `--.   .'.'                  ,'   `. :_,"  `.
 ,.   ;   .   \                ,'     |         `.
' :  :    |   `.             ,'       |\         `.  _
 `.   ._  |     \         _.'         | .      ___ `" :
        : '    . \      ,'  .         ' |     :   `...'
       ,'  \      `.   .            ,'  |     '  __
      .    `.      |    \          .'   '    .  (  `.
    .'      \`.___,'     `-.____.-'    '     :   `-.'
     .   ,". \ ..___             _    /      :    .
     :   . :  \|/\ `"'--------+"|,' ,'       `-..' :
      `-" .'  `: `"-.._______,.\|  .'              '
          `--. _ `._            _,'       ,""-.__,'
             " :   `"--.....--"'    __   .
             ,-'                ,.-"  `-'
             :  ,..             .   ,"".
            .'  .  :   __..._  `"-. :   :
           `.._  : ' ,'      `"--..' `--"
               `-' `" mh
```

Gastly functions as a bash version of GRC’s ShieldsUP!, yet also corrects open ports by hiding them. If the user elects to run it, the script checks the machine for open ports with ICMP pings and stores the information in three files. If the user chooses to filter the reported ports, the script finds the open ports stored in one of the report files, creates a new report file, stores the ports there, and then the script retrieves each storing it as a variable. The script then creates iptables rules based on these ports to still allow outgoing traffic but block incoming.

The functions of Gastly is debated to be a violation of firewall functionality and that GRC misunderstands the reasons why ports are open on non-Windows machines. Therefore, Gastly should be treated as experimental and a possible failure. If the system hardening guide functions properly, Gastly should appear broken anyway as Cloyster has blocked port scans and adds any IP conducting a port scan against the localhost to a blacklist file for 24 hours. However, that's assuming the user actually entered 'y' when Cloyster was ran.

Each script can be done whenever, but the order was chosen as some follow the steps of other like Cloyster and Haunter follow Metapod for blocking access to SSH and Cron.

Evaluation

Each script was run multiple times to test each option available to ensure every command worked. Each time there was a tweak, the script modified would be set to run the changed function to ensure it worked. At the end of the day, all of the scripts would be tested again to make sure no changes broke the script system.

Script testing:

BASEC was tested for the functionality of the navigational control bar, the intermission pages between pages that would run scripts, installations, the next page based on user input.

Metapod was tested for functionality of each command from updates to disabling the root account. There was a problem when the user shell account when all 10 system hardening scripts were 1 and when the user was prompted to basically disable their own account. The guide the script was based off of didn't specify that it's use is meant for system administrators to disable inactive user accounts, which wouldn't be necessary for new users.

Haunter was tested whether or not the user account was permitted back to Cron while everyone else was still denied. This script was one of the easier scripts as it functioned perfectly.

Grimer was tested for its functionality of disabling and / or removing software tools not commonly used. One of the easiest ways of doing this was to check each file per command, check permissions levels for some of the tools disabled, and check through aptitude or apt-get that software packages were uninstalled.

Koffing was tested by opening up user settings to change the user avatar by taking a picture of myself to see that the webcam is disabled and by opening sound settings to test input volume only for it to fail to pick up any sound. Disabling the mic via sound settings isn't full proof and can be easily enabled again.

Machamp was tested by watching the root directory and the terminal with /tmp being moved. Testing on a system with FDE has failed since the specific FDE tool wasn't used to do so. Testing on a freshly installed system worked great, but still proved that the user should set up a /tmp partition during the install process.

Muk was tested the same exact way as Grimer. As previously mentioned, the only reason these two scripts aren't the same script has to do with Grimer removing tools that affects SysSec and Muk removes tools that affects NetSec, but both still remove unused tools not commonly used in an identical fashion. Muk, too, was one of the easier scripts to test and all it took to prove the effectiveness was check with aptitude or files changed. Since Muk follows Grimer, when testing Muk, processes were also checked to make sure no daemon was running that utilized the tools. Any that were still active were not once a reboot was conducted.

Weezing was tested by checking network settings and probe attempts from other machines. A second computer was used to portscan the former IP address of the tested system and the wireless router was checked to view any traffic from this IP address or MAC address. Bluetooth was also checked by attempting to pair with it, looking for active devices in range, and checking Bluetooth settings. Like Muk, once finished, processes were checked to make sure these wireless capabilities were disabled.

Charizard was initially tested by making sure the IPSEC rules have been added. This function has never failed. Then, the firewall rules for ufw were checked to make sure changes have been made. A light test of whether or not ufw was working correctly was conducted and proven successful. The reason a full test on ufw wasn't conducted was due to my several years of experience using ufw having tested many times its functionality. Last, the ufw rules regarding HTTP were verified to have been added to ufw rules while ensuring traffic was still active.

Cloyster was tested by checking iptable rules, attempting connections, attempting port scans, with and without NULL packets, and getting locked out successfully (as previously stated, anyone conducting a portscan is blacklisted for 24 hours), and checking sshd for all the rules that were supposed to be added. They were added successfully.

Gastly has been tested the most for previously mentioned reasons of it being part of my original SIP. The nmap command works perfectly every time, the port retrieval command only had to be tested once to make sure it works, and the iptables commands were tested multiple times to ensure they work. The iptables part had more to do with what I wanted to output and how firewall rules were saved more than the actual iptable commands.

Not all errors are listed here. Every time a script was saved, it was tested on the changes made. If an error was produced, then the code was modified. Errors broke the scripts. What I can say is that they were tested myriad times in multiple ways to make sure they functioned **exactly** as intended. More time went into testing than actual coding since BASEC is a system of scripts and not a solitary script.

- **Simplicity –** Almost all scripts are run where the user can understand the purpose from BASEC, the function from script documentation and watching the scripts in action, by outputting whatever output a command has in a different color. Otherwise, each script states when each command is complete. Users will know which ports to hide as per Gastly's function and which software to disable and / or remove by stating what tools are disabled and / or removed before the user inputs to do so. Users are warned what functions they may want to skip by warning messages stating, in detail, why they may elect to skip a function. Some functions are necessary for basic security, but not basic enough to be easily understood by new users. Last, the users can adopt the same method of running BASEC to run each of the complementary scripts and are told so in BASEC.

- **Effectiveness –** Effectiveness and functional have been hard to combine for network security as securing the firewall doesn't mean it still allows network activity. Apart from the networking aspect, all of the scripts and their functions have proven to be effective with their purpose. As for networking, additional testing and debugging was needed to ensure the firewall still secures the machine while allowing network activity. This has been the most complicated part of BASEC.

- **Time / Resource Usage –** Excluding Machamp, all of the scripts are lightweight and very quick at their functions. Even Gastly is fairly quick and provides feedback on its current status while the user waits. Machamp still requires a lot of resources and time to complete, but no amount of coding will change this function. At least the user is warned about Machamp being time- and resource-consuming.

- **Practicality –** The scripts are unique enough amongst each other that each script still serves a practical purpose for their intended use. Muk and Grimer are fairly similar, but, as previously stated, they are separated due to their tools affecting two different chapters of BASEC. The intended use of the scripts already practical, thus why they have been implemented into BASEC. Only Gastly serves as not being possibly practical due to the debate of whether or not Linux ports should be open or filtered.

- **Functional –** The entire purpose of testing was to make certain that each script was functional. A broken script isn't useful to the project. More time was spent testing than coding for this exact criterion. The biggest issue had to deal with firewall rules functionality breaking the functionality of networking so more tweaking and testing was needed for functional and effectiveness criteria not to break each other.

- **Diversity –** All of the scripts are written and documented well enough that users can easily modify the scripts with their own functions. As long as the commands they enter actually work and they follow the scripts' template, modifying the scripts should be easy for new users. It was coded in such a way that those new to coding would have an easier time reading the scripts than just writing the scripts without functions.

- **Organization –** The code within the scripts are well-organized enough that a user not watching their home directory would not know that temporary files were created or downloaded in script before said script deletes them. The scripts themselves are well organized based off of BASEC and BASEC is well organized to secure each type of security prior to network security in case any issues every arose from the chapter's scripts.

- **Stability –** The scripts have been tested weekly to make sure they are still up-to-date in case any other update may crash the scripts. This has happened to Autosnort Debian, which is the only reason why BASEC is intended for Ubuntu and Ubuntu-based distros use and not the full Debian / Ubuntu tree. The coder behind the Autosnort scripts has not updated Autosnort Debian breaking it in so many sections.

- **Uniqueness –** Uniqueness and practicality have blended together. The scripts are unique enough to be separate from one another, but they are also unique enough that the user may want to build on them. Instead of using ICMP scans or filtering open ports, Gastly can be modified to use a series of packet types to look for open ports, coded to compare and contrast the results, and close which ever ports were open to one packet type and filtered to another. Koffing can be coded as a daemon to enable webcam and mic when on the system is on a different network (as if stolen) to record information and then disable both when on the home network. Each script is unique and practical in their own way and at least half can be modified to be their own SIPs (I've lost count of SIP ideas I've created thanks to BASEC after the 17th).

- **Cross-platform –** For sake of trying to complete the Linux support of the scripts, testing for other platforms has been permanently shelved. The primary concern is that BASEC and the complementary scripts function perfectly on Linux.

- **Additions –** Adding more scripts can increase the effectiveness and practicality of BASEC. Autosnort is a perfect example. It's a simplified snort install script that increases the effectiveness of basic security by easily installing a powerful IDS where the user needs minimal input to run and a little bit of understanding of networking to use the IDS to its intended purpose. Any additional scripts created will be tested and added where necessary throughout the Innovation Brief.

Project Completion Assessment

BASEC and the complementary scripts are great for basic security and has established a framework for expanding on that security. However, each new SIP idea BASEC generates makes the current version feel lackluster by how much more can be done, even though those new ideas crosses the threshold from basic security to advanced security. The issue is that BASEC was written in many ways for myself on a functional level and for a fellow classmate of the simplicity level. The parts written for myself don't appear as great since I come up with new ideas that advance the security to my own level of understanding that may not be the least bit necessary for new users like the Project pages. It's difficult to provide an objective assessment of what I want from what is needed for new users, especially when certain scripts are incorporated that may never be of use to new users like Autosnort.

Excluding cross-platform, BASEC has definitely met my standards and some parts have even exceeded those standards. I'm comfortable with dropping the cross-platform aspect for various reasons. The scripts can be modified to work on UNIX or other Linux distros and coding the scripts for Windows would be a lost cause. Windows at its most secure is still fairly insecure and there's no guarantee that Windows addition of bash will run the scripts, let alone allow them to change the system configurations. BASEC as a PDF that just teaches the user security and what tools they can use on Windows still exists; only the complementary scripts are null and void. As previously mentioned, the additions aspect is both exciting and frustrating in trying to separate what I want for my own use and what I want for new users to use that they may actually use. The Koffing modification I mentioned under Uniqueness is an example of what I want, but what a new user may not use for so many reasons: 1) if the users use FDE and don't use Project Honeydrive, then the script is irrelevant. 2) If the users don't use FDE or Project Honeydrive, then the script risks being made useless by thieves thinking Wi-Fi is broken without knowing how to fix it. 3) If the system is stolen by a smart thief that knows to remove the drives and put them into a system with write-blocking, then the script is irrelevant. There's are all matters of more advanced security that a new user may not bother with, especially setting up FDE or setting up an additional drive to act as a honeypot OS (Project Honeydrive).

BASEC only had **one goal**: create a user-friendly, interactive guide that teaches security, allows navigation, hardens the system, and adds additional security layers. This one goal had four objectives.

First Objective

The first objective was to create a guide that teaches users the basics of security using PerSec, SysSec, and NetSec as the three pillars of focus. This objective had three activities of researching tools and methods advised from myriad sources, write own advice from a beginner and experienced point of view on top of the professional advice, and try to provide at least two tools per each page so the user has some choice. The intermission page chooses a tool depending on the page and asks the user if they want to install the downloaded tool. The professional advice tended to be solid but also technical jumping over the heads of new users (speaking from personal experience). Writing the personal perspective was actually a great thing that acted as a translator for new users so that they may understand the concepts taught about without having to get technical yet. The only issue with this objective is that this proved to be the most time-consuming aspect of the entire SIP (not to be confused with tweaking and testing functionality vs effectiveness of scripts being more time-consuming than actual coding). There's so many professional opinions worth seeking, many of which disagree with each other on some level so sifting through which information was essential and adding my own opinion made it that much more involved.

Six Hats

- **White –** This objective has been the most time-consuming element of BASEC. It could not have been shortened without cutting corners, thus violating the integrity of the guide. Most aspects of security and system administration has more than one tool to fulfill its need. Each page of BASEC has successfully followed this by listing at least two software options per each aspect.
- **Red –** The time-consuming element has been frustrating, but relieving and hopeful after completion. I feel a modicum of pride in the thoroughness and easy-to-follow teachings of each page.
- **Black –** BASEC may be made obsolete tomorrow by any professional releasing their own interactive guide that's more thorough and carries the name of known professional.
- **Yellow –** If no professional does make BASEC obsolete, then I can expand BASEC to be fully cross-platform, change it from a script system to a PDF and a single program, and expand the security aspects beyond what they are currently. Even if it is made obsolete, BASEC still provides some value in teaching new users basic security from someone who has experienced their frustrations a lot more recently than the professional.
- **Green –** See Yellow.
- **Blue –** There's no time to exploit the Yellow / Green hat of going cross-platform with a PDF and program. Focus on managing the current script system and only carry out the Yellow / Green hat once the SIP fair is complete.

BASEC is **not** innovative at this level standalone.

Second Objective

The second objective is to actually make use of the former SIP Cloaks&&Daggers. This objective is a direct copy from the past Innovation Brief with the first five activities for creating a script that installs Nmap, uses Nmap with ICMP packets, stores the open ports in a temporary results file, retrieves those open ports, and changes the iptables rules for the ports to drop incoming traffic. The sixth activity informs users that the changes won't take effect until their next login session so running the script twice in one sessions is fruitless. The last activity displays the next page to read since running a script that execute a script. Using Cloaks&&Daggers and renaming it Gastly was a good idea since it still provided some functionality and on an elementary security level since we learn of GRC's ShieldsUP! in the security fundamentals class. This script only took a day to code and test.

Six Hats

- **White –** This objective was just an adaptation of an existing, functional script. The only changes made are activities six and seven.

- **Red –** I felt it was a good idea to make use of the script and add more functionality to BASEC. I'm glad I added it as it's proven useful in the past and it was one of the easiest aspects of the SIP since it was already created and tested in the past.
- **Black –** It may or may not violate the purpose of open port statuses possibly leading into problems of its own in the future.
- **Yellow –** If used, the user knows that they're invisible to attackers (especially due to Cloyster).
- **Green –** Gastly can be coded with more experience and knowledge of firewall ports to know which ports to avoid trying to filter. It can also be coded to do the same for the router, but doing this violates the purpose of the SIP BASEC of only securing the users' machine and allowing them to take security into their own hands from there.
- **Blue –** If there's time, definitely research firewall functionality more in-depth, find the ports that should never be filtered, and add an if statement for Gastly to exit if the port variable matches any of these researched ports.

Gastly (Cloaks&&Daggers) is **not** innovative at this level standalone. It was designed to work with three scripts for penetration testing one's own firewall security, but I perceived it to not be innovative enough. This led to the idea of BASEC, which, in turn, led to the merging of both SIPs.

Third Objective

The third objective is to create a script that conducts in system hardening for the user since, as new users to security and / or Linux, they might not know how to harden their own systems yet. The eight activities are to have one function that updates the system and adds unattended upgrades, install widely used security tools, secure the terminal, permissions, and memory, secure access to Cron and SSH, create port knocking for SSH, create partition and move /tmp, disable and remove services not commonly used or inherently insecure, and add IPSEC and firewall rules. This objective was a bit time consuming due to the last activity mostly, but also testing each activity. The script since then has been broken down where each activity may be its own script or two. Excluding Gastly, this objective is now known as the nine complementary scripts. The script initially only took two days to code and test, but has since been tweaked and tested irregularly since then. The final activity of Objective 2 wasn't added to Objective 3. Again, the third objective should have been made into its own goal making Objective 4 its own goal as well. While the activity to add the next page the user should view wasn't written in the Innovation Brief, it was coded into each script.

Six Hats

- **White –** This objective has been the second most time-consuming element of BASEC. If the initial Innovation Brief were written at this point, this objective would be broken down into multiple objectives and made into its own goal. Some of the scripts are not essential, not basic, or would otherwise be ignored. However, they still offer important aspects of security that merits their existence and attachment to BASEC. Even if new users don't want to disable Wi-Fi, it's undeniable how wireless networking is less secure than wired networking inherently and can be easily mitigated.

- **Red –** I feel confident about these scripts proving their worth to this project and could add more. In all honesty, more script ideas have been added while finishing the Innovation Brief and that makes me feel very confident about the framework established by BASEC to cover so much of basic or intermediate security.
- **Black –** There could be more scripts taking care of more aspects of basic security.
- **Yellow –** BASEC has established a framework where Black will always be relevant because there's always going to be more that can be added to BASEC, which, in turn, keeps BASEC constantly relevant. There's nowhere to go but up from here.
- **Green –** I could actually write a framework using each page of BASEC and write ideas for each page of how users with basic or intermediate understanding of security and Linux may apply these ideas, add to them, or tweak them for their own purposes.
- **Blue –** If there's time, at least establish an outline for the framework. Wait until the SIP class is finished to write it and try to add it to BASEC before the SIP fair. Don't add it to the actual BASEC script but an extra document for the user to peruse through.

The nine scripts that formerly made up Metapod are **not** innovative standalone or combined. These scripts are based off of CyberPunk's System Hardening Guide and coded for script use and witch simplicity for new users to understand instead of the raw commands that make the guide of which the scripts are based off. At best, I just simplified CyberPunk's work and made it more versatile, organize, and compartmentalized.

Fourth Objective

The fourth objective is to create a script that houses the guide of Objective 1 and controls the scripts of Objectives 2 and 3. The activities were to add the guide to a script (making it interactive), separating each page to its own function (making a table of contents functional for jumping through sections of the guide), and adding intermission pages between each specific page to run one of the scripts relevant to the previous page or to skip the use of that script moving to the next page. The last two activities are adding commands to the intermission pages like downloading and installing bleachbit, pwd.sh, or Autosnort when I haven't created any scripts to do the same function and adding a navigational control bar that makes use of the table of contents, each page being its own function, and allowing the user to quit the script. This took an undiscerned amount of time less than a week to code and test.

Six Hats

- **White –** This objective has been a very critical aspect of the project since each objective standalone is not innovative. It combines them all to make them fairly innovative and an idea that, surprisingly, has not been explored. BASEC is a whole is a simple but powerful concept. The navigational control bar is not intrusive, easy to use, and easy to understand. The color coding of the bar and headers allows users to know to read the script just like an actual guide. The display commands are not great. This was coded to be displayed in a tiled terminal of a specific resolution and isn't adaptive to display settings.

- **Red –** This objective really brings the SIP together and wraps it nice and neatly. I feel very confident about every aspect of it, excluding the display issue previously mentioned.
- **Black –** See White about display issue. This really only affects sections that failed to be only a page long.
- **Yellow –** Excluding the display issue, this objective is easy to understand and execute. It makes the guide and project as a whole look fantastic. If the display settings adjustment can be added so that there's more text that can be added, then it would be perfect!
- **Green –** I could code a way to detect resolution and modify how much text is displayed. How would I treat the text though? Would it be one giant pool of text?
- **Blue –** Research resolution detection as it has more value than the mentioned issue. However, don't spend time actually making adjustments. The script was coded to display how it does since the use of other display commands either failed to function properly or required an additional file to display the contents. That means each page wouldn't be a function of the BASE script but would each be one of many txt files in a folder of BASEC pages for the BASEC script to call and display. This is more troublesome than valuable.

BASEC the script is **not** innovative standalone. However, combined with all of the other objectives, BASEC the SIP is **definitely** innovative. It's a simple but powerful concept that hasn't been explored for some reason. It's a guide that doesn't just teach users the basics of security easily enough for them to understand but it also secures their systems for them. It's a system hardening project that doesn't just secure their systems for them but also does so in a simplified, well-organized, and documented way that the user understands what's happening. This is especially true since each script that hardens their system follows a page that teaches them about it!

I've created an analogy that best describes the mentality of BASEC as a whole. Give a man a fish and you feed him for a day; teach a man to fish and you feed him for a lifetime. Secure a user's machine and you secure him until an update or new installation; teach a user security and you secure his / her systems for a lifetime. The problem with this model is that it elects one over the other when we should use **both**! Give a man a fish to feed him for that day and spend of the rest of the day teaching him to fish. Teach a user the basics of security and then implement those security measures to secure the individual until they are knowledgeable and experienced enough in Linux and security to secure themselves. If we stopped upholding the definition of insanity by constantly teaching anyone security measures with the assumption that they may have actually understood anything or want to keep themselves secure, then we could see the problem in a whole new light. People have made it clear that they don't understand and / or don't care about security. That means it's our responsibility as security professionals to uphold security for them. Security companies could make contracts with operating system developers to ensure a system is hardened before release or security professionals could go as far as automating and simplifying something like system hardening. Security is a lot like our health. We could vaccinate ourselves and use herd immunity to keep ourselves secure or we could not and allow malware and cyber-attacks to become rampant. The problem is that we've chosen the latter. On the fundamental level, I've decided to help vaccinate new users to protect them and the herd they affect.

APPENDICES

Appendix A – References

Burns, B., Killion, D., Beauchesne, N., Moret, E., Sobrier, J., Lynn, M., ... Guersh, P. (2007). *Security Power Tools* [Kindle]. Retrieved from Amazon.com

cogNiTioN. (1999, Dec. 30). Intro to Cron. *UnixGeeks.org*. Retrieved from http://www.unixgeeks.org/security/newbie/unix/cron-1.html

Ferrara, J. (2012, Sep. 18). Why Most Cyber Security Training Doesn't Work. *Wombat Security Technologies*. Retrieved from https://www.wombatsecurity.com/about/news/why-most-cyber-security-training-doesnt-work

Gibson, S. (2003, Oct. 6). Internet Connection Security for Windows Users. *Gibson Research Corporation*. Retrieved from https://www.grc.com/su-explain.htm

Ho, Erica. (2010, Apr. 07). ShieldsUP Tests Your Firewall for Vulnerabilities. *Lifehacker*. Retrieved from http://lifehacker.com/5511734/shieldsup-tests-your-firewall-for-vulnerabilities

Introduction. (n.d.). *Nmap*. Retrieved from http://nmap.org/

Nmap from Beginner to Advanced. (n.d.). *Infosec Institute*. Retrieved from http://resources.infosecinstitute.com/nmap/

Pagliery, J. (2014, Jun. 26). 5 online privacy tips from an ex-FBI agent. *Cable News Network*. Retrieved from http://money.cnn.com/2014/06/26/technology/security/fbi-privacy-tips/

Raspberry Pwn: A pentesting release for the Raspberry Pi [Web log]. (n.d.). Retrieved from http://blog.pwnieexpress.com/post/24967860602/raspberry-pwn-a-pentesting-release-for-the-raspberry

System Hardening Guide. (n.d.). *CyberPunk*. Retrieved from https://n0where.net/system-hardening-guide/