In but not Out: Protecting Confidentiality during Penetration Testing

GIAC (GSEC) Gold Certification

Author: Andrew Andrasik, andrew@cy-int.com
Advisor: Stephen Northcutt

Accepted: August 19, 2016

Abstract

Penetration testing is imperative for organizations committed to security. However, independent penetration testers are rarely greeted with open arms when initiating an assessment. As firms implement the Critical Security Controls or the Risk Management Framework, independent penetration testing will likely become standard practice as opposed to supplemental exercises. Ethical hacking is a common tactic to view a company's network from an attacker's perspective, but inviting external personnel into a network may increase risk. Penetration testers strive to gain superuser privileges wherever possible and utilize thousands of open-source tools and scripts, many of which do not originate from validated sources. Penetration testers may gain access to all compartmented sections of a network and document how to repeat successful exploits while saving restricted data to their laptops. This paper illustrates secure Tactics, Techniques, and Procedures (TTPs) to enable ethical hackers to complete their tests within scope while reducing managerial stress regarding confidentiality. A properly conducted independent penetration test should provide essential intelligence about a network without jeopardizing the confidentiality of proprietary data.

1. Introduction

In practice, confidentiality prevents information disclosure to unauthorized individuals (McFarland, 2014). However, security controls, such as the National Institute of Standards and Technology Special Publication 800-53, require external entities to test the confidential nature of networks and document compliance (NIST, 2013). Managers and administrators must duplicate rigorous security measures for independent assessors or hinder advanced security testing (NIST, 2013). Some confidentiality measures require monitoring all software and hardware interactions in and around a network through strict regulation (FIPS, 2006). In these environments, regulatory restrictions may interfere or prevent the use of generic penetration testing tools, like Kali Linux's basic build because of undisclosed code origination (FIPS, 2006). Reducing the size and scope of the base build is the first step to meeting these requirements.

In all of these cases, monitoring networks include every part of any information system or device on employees and users that may inadvertently attempt to communicate (FIPS, 2006). The security industry can modify existing technologies to create a custom platform that will facilitate compliance and security concerns. Using a custom platform instead of a one-size-fits-all solution will:

- Provide testers with a more efficient and effective testing platform,
- Eliminate unnecessary tools that may weaken security,
- Ease compliance concerns of managers and system administrators, and
- Facilitate monitoring concerns regarding tools used by the independent assessor.

1.1 The Need for a Customized Testing Platform

A unique networking environment requires a specialized platform to complete a security assessment. Popular penetration testing operating systems (OSs) may not always provide full disclosure for applications and scripts included in the OS's base build. Kali Linux, maintained by Offensive Security, is considered the de facto penetration testing OS used by ethical hackers (Scott, 2016). Offensive Security created Kali by using a Debian-based Linux distribution and adding over 600 penetration testing and forensic tools (Offensive Security, 2016a). If an organization requires monitoring all software entering its network, Kali Linux may be more of an obstacle than a solution for penetration testers walking in with a base build. To detangle this complication, a penetration tester can create a disposable ISO image that will not jeopardize confidentiality during an assessment.

Creating a custom ISO image can solve these dilemmas without limiting tools available to a penetration tester. The goal of creating a custom ISO image is to build a fully functional advanced penetration testing OS that penetration testers can easily use and discard on an engagement. An ISO image can mount to a system and run independently of a hard drive. ISO images can include proprietary licensing information and stockpiles of open-source information to assist during a penetration test. The small size and intuitive nature of moving a single file help with redundancy and increases the number of stable backups a penetration tester can use without hesitation. With all of these advantages, ISO image files significantly amplify a penetration tester's arsenal, but what exactly are ISO files and how should a penetration tester exploit their advantages?

An ISO image is an archive file first introduced by the International Organization for Standardization (ISO) under ISO 9660:1988. ISO 9660 created a standard for Compact Disc Read-Only Memory (CD-ROM) by outlining the file structure, volume attributes, and process for input and output of data streams mounting to an OS as a volume (ISO, 2015). In the 1990s, ISO files evolved as a universal disk format (UDF) standardized by ISO and the International Electrotechnical Commission that dramatically increased functionality, applicability, and maximum file size (Library of Congress, 2012). Currently, ISO files are extremely versatile. ISO and the European Computer Manufacturers Association (ECMA) introduced multi-extent (fragmentation) capabilities in ECMA-167 Level 3. These new capabilities removed file-size limits to cover new technologies, which now include Blu-Ray Disks or whole file structures (ECMA, 1997). The custom ISO image selected for penetration testing is designed to meet ISO Level 2 specifications that limit the custom image to 4.2 gigabytes (GB). This is by design so the custom image can easily fit on a one-time writable digital versatile disc DVD or a universal serial bus (USB) drive if permitted.

Penetration testers require access to every testing tool necessary to perform an effective and efficient test. With a maximum file size of 4.2 GB, testers realize the importance of selecting the best host OS for the custom ISO image. External penetration testers frequently move to various networks that utilize different technologies. Kali Linux is a clear choice for the base OS if we remove the scripts and applications that are not essential for operations. We can always add or subtract tools to the custom distribution after performing due diligence to verify the source, integrity, and necessity of the code. Offensive Security made Kali 2.0 a rolling release based on Debian Jessie, which regularly incorporates code from Debian testing (Offensive Security, 2016b). Section 3.1, Version Control, provides special instructions for maintaining a custom ISO image's integrity.

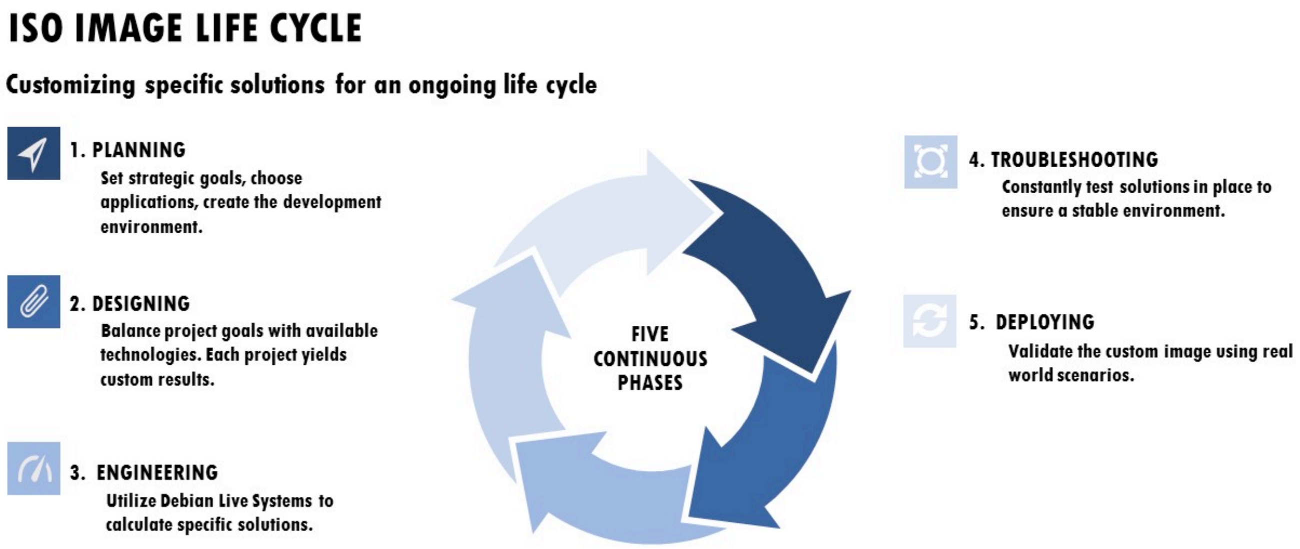
If an organization needs to test less common protocols or technologies, it helps if a penetration tester has a host OS that can communicate directly with the computer without a hypervisor. Virtualization increases efficiency in many ways but specifically with wireless technologies, including Bluetooth, installing the OS directly on a host increases stability and functionality. Running Kali Linux as a virtual machine may deny access to wireless drivers on the host machine. Using an external wireless adaptor, such as the Alfa AWUS036H, utilizes previously existing wireless drivers while increasing range and speed (Sak, 2016). Even in extremely sensitive areas, a penetration tester should have access to wireless assessment tools. If an organization bans wireless signals throughout the topology, wireless equipment and drivers are used solely to confirm that wireless communications do not exist.

2. Creating the ISO

In *A Down-to-Earth Guide to SDLC Project Management*, Joshua Boyde explains how to adapt the system development life cycle (SDLC) to accomplish projects using the Project Management Body of Knowledge (PMBOK). An enhanced version of Boyde's life cycle decreases development time using PMBOK and tailors the process specifically for ISO development; steps shown below:

1. Planning,
2. Designing,
3. Engineering,
4. Troubleshooting, and
5. Deploying.

After creating a functional, custom ISO image, the same continuous cycle assists in automating repetitive processes and incorporating version control.



ISO IMAGE LIFE CYCLE

Customizing specific solutions for an ongoing life cycle

1. PLANNING
Set strategic goals, choose applications, create the development environment.

2. DESIGNING
Balance project goals with available technologies. Each project yields custom results.

3. ENGINEERING
Utilize Debian Live Systems to calculate specific solutions.

4. TROUBLESHOOTING
Constantly test solutions in place to ensure a stable environment.

5. DEPLOYING
Validate the custom image using real world scenarios.

Figure 1: ISO Image life cycle

2.1. Planning

The planning phase includes reviewing the mission statement, selecting applications, and creating the development environment. When creating a production-grade custom OS, a virtualized environment is vital. Starting software includes OS X El Capitan version 10.11.6 with VMware Fusion Professional Version 7.1.3. Starting with a stable, basic build is important. Currently, Kali Linux 64-bit, version 2016.1 with a SHA1 Sum “deaa41c5c8f26b7854cafb34b6f1b567871c4875” is available for download from https://www.kali.org/downloads/. To start, set up the Kali virtual machine with at least 100 GB of storage to allow numerous packages to expand during ISO creation and as much random access memory (RAM) as the host systems allows.

When using multiple OSs, it is important to clarify which is OS is used during each step. The OS on the host machine is titled host OS. The virtual machine running Kali Linux used as a development environment is the Kali Development Virtual Machine (KDVM). The latest ISO image build used to test features from these customization processes is the Testing Virtual Machine (TVM). Delete the TVM after testing each iteration.

The KDVM requires VMware tools and Kali's dist-upgrade using the following commands:

```
apt-get update
apt-get dist-upgrade
apt-get install open-vm-tools-desktop fuse
reboot
```

VMware tools incorporate seamless interaction between the host OS and the virtual OS while adding full-screen capabilities. Installing either version of VMware tools should work, but Offensive Security recommends using the open-vm-tools-desktop package (Offensive Security, 2016b). Using Aptana Studio 3 as the default text editor helps clarify file locations during customization.

Aptana Studio 3 is a powerful web text editor that incorporates most standard integrated development environment (IDE) features (Appcelerator, 2014). Aptana Studio 3 is ideal because it has a small number of dependencies, runs on most OSs with Java, and rarely crashes. Aptana Studio 3 displays the file structure of the development environment on the left-hand side, clarifying file locations while customizing the build. When a user is comfortable using Aptana Studio 3, programming languages such as Ruby can automate processes during ISO creation and utilize the built-in terminal. The latest version is Aptana Studio 3, build 3.6.1.201410201044.

After the KDVM is set up and rebooted, it is time to create the first iteration of the TVM. Steps for creating a standard Kali Linux ISO image include the following:

```
apt-get install curl git live-build cdebootstrap
git clone git://git.kali.org/live-build-config.git
cd live-build-config
./build.sh --distribution kali-rolling --verbose
```

Microsoft Office User

Executing the commands mentioned above on the KDVM creates an ISO image titled "kali-linux-rolling-amd64.iso" in the folder /live-build-config/images/. It is important to ensure the ISO image is fully operational by installing it using VMware Fusion as a new virtual machine. If the TVM ISO image installs a new OS that functions in the same manner as the KDVM, move to the next phase. If the TVM does not function properly, repeat the life cycle by troubleshooting the ISO image creation process and testing it in VMware Fusion. Section 2.4, Troubleshooting contains more details on problem-solving and the development life-cycle.

2.2 Designing

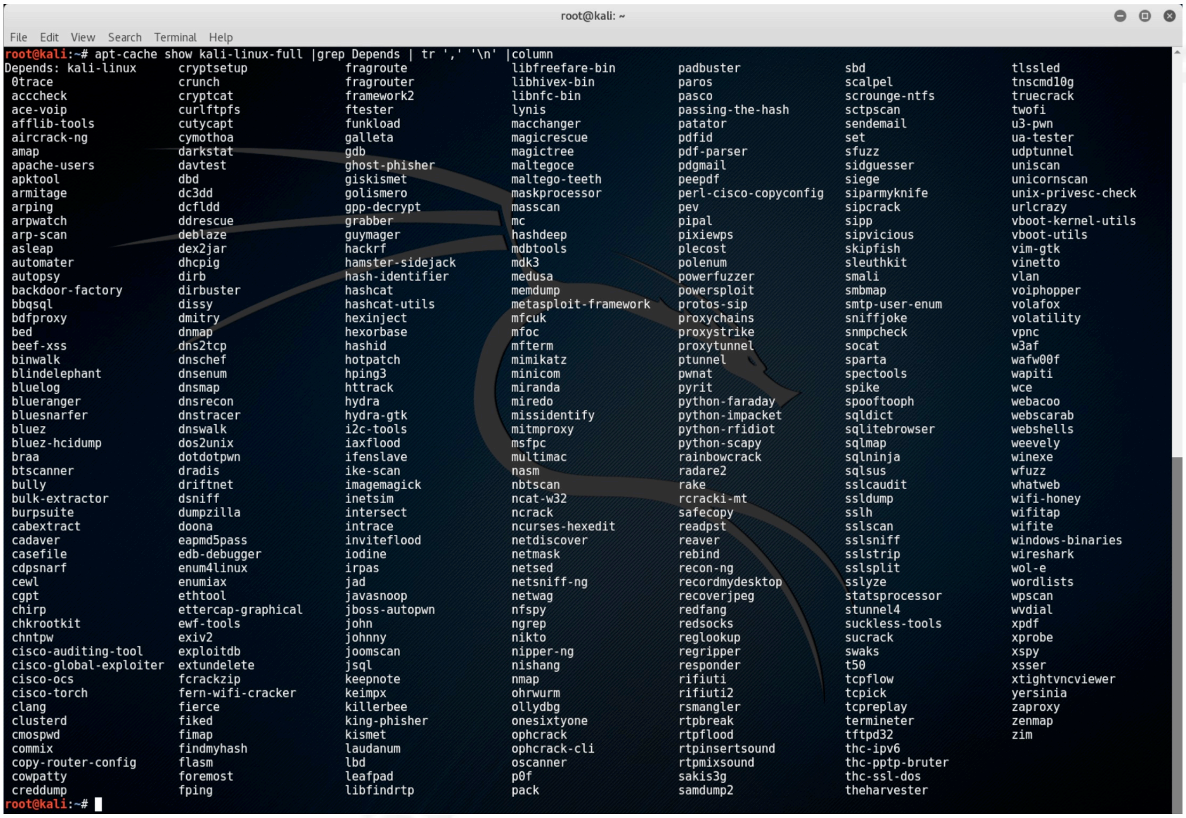
The design phase balances the project mission with specific applications and functions to exceed project expectations. Developing a custom ISO image is different for each use case. This design centers on performing an advanced penetration test without a hard drive or internet access. The custom ISO runs directly from the RAM on a laptop. All intelligence required for hacking information systems, licenses for proprietary tools, and necessary scripts are in one compressed ISO image. Running an OS from a live state increases the chances of crashes and losing data; thus, stability is a high priority.

Empirical evidence suggests the more applications running on an OS directly correlates to the stability of the OS running in a live environment. However, all packages installed on Kali Linux are selected with the purpose to aid security assessors during a penetration test. The goal is to design a lightweight, stable penetration testing OS that can run in a live environment. The apt-cache command used in the example below assists in determining the original state concerning packages. The apt-cache command displays information stored by Debian's package manager, APT's internal database. APT's database updates during the apt-get update operation (Debian, 2015). For more advanced operations, the "apt-cache pkgnames" command displays a single file listing of all packages that appear at least once in APT's cache. For these purposes, the following command lists most of the independent applications Offensive Security added to Kali Linux's distro, specifically the kali-linux-full package.

```
apt-cache show kali-linux-full |grep Depends | tr ',' '\n' |column
```

Microsoft Office User

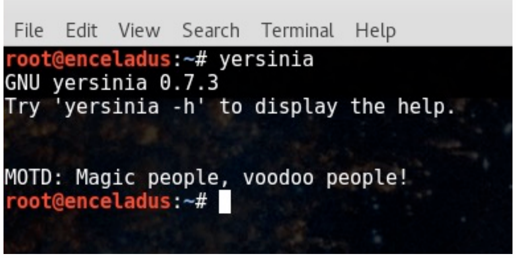
The following screenshot lists the current packages on the first iteration of TVM using the default Kali Linux Gnome Kali-Linux-Full build.



```
root@kali:~# apt-cache show kali-linux-full |grep Depends | tr ',' '\n' |column
Depends: kali-linux     cryptsetup              fragroute               libfreefare-bin         padbuster               sbd                     tlssled
 0trace                 crunch                  fragrouter              libhivex-bin            paros                   scalpel                 tnscmd10g
 acccheck               cryptcat                framework2              libnfc-bin              pasco                   scrounge-ntfs           truecrack
 ace-voip               curlftpfs               ftester                 lynis                   passing-the-hash        sctpscan                twofi
 afflib-tools           cutycapt                funkload                macchanger              patator                 sendemail               u3-pwn
 aircrack-ng            cymothoa                galleta                 magicrescue             pdfid                   set                     ua-tester
 amap                   darkstat                gdb                     magictree               pdf-parser              sfuzz                   udptunnel
 apache-users           davtest                 ghost-phisher           maltegoce               pdgmail                 sidguesser              uniscan
 apktool                dbd                     giskismet               maltego-teeth           peepdf                  siege                   unicornscan
 armitage               dc3dd                   golismero               maskprocessor           perl-cisco-copyconfig   siparmyknife            unix-privesc-check
 arping                 dcfldd                  gpp-decrypt             masscan                 pev                     sipcrack                urlcrazy
 arpwatch               ddrescue                grabber                 mc                      pipal                   sipp                    vboot-kernel-utils
 arp-scan               deblaze                 guymager                hashdeep                pixiewps                sipvicious              vboot-utils
 asleap                 dex2jar                 hackrf                  mdbtools                plecost                 skipfish                vim-gtk
 automater              dhcpig                  hamster-sidejack        mdk3                    polenum                 sleuthkit               vinetto
 autopsy                dirb                    hash-identifier         medusa                  powerfuzzer             smali                   vlan
 backdoor-factory       dirbuster               hashcat                 memdump                 powersploit             smbmap                  voiphopper
 bbqsql                 dissy                   hashcat-utils           metasploit-framework    protos-sip              smtp-user-enum          volafox
 bdfproxy               dmitry                  hexinject               mfcuk                   proxychains             sniffjoke               volatility
 bed                    dnmap                   hexorbase               mfoc                    proxystrike             snmpcheck               vpnc
 beef-xss               dns2tcp                 hashid                  mfterm                  proxytunnel             socat                   w3af
 binwalk                dnschef                 hotpatch                mimikatz                ptunnel                 sparta                  wafw00f
 blindelephant          dnsenum                 hping3                  minicom                 pwnat                   spectools               wapiti
 bluelog                dnsmap                  httrack                 miranda                 pyrit                   spike                   wce
 blueranger             dnsrecon                hydra                   miredo                  python-faraday          spooftooph              webacoo
 bluesnarfer            dnstracer               hydra-gtk               missidentify            python-impacket         sqldict                 webscarab
 bluez                  dnswalk                 i2c-tools               mitmproxy               python-rfidiot          sqlitebrowser           webshells
 bluez-hcidump          dos2unix                iaxflood                msfpc                   python-scapy            sqlmap                  weevely
 braa                   dotdotpwn               ifenslave               multimac                rainbowcrack            sqlninja                winexe
 btscanner              dradis                  ike-scan                nasm                    radare2                 sqlsus                  wfuzz
 bully                  driftnet                imagemagick             nbtscan                 rake                    sslcaudit               whatweb
 bulk-extractor         dsniff                  inetsim                 ncat-w32                rcracki-mt              ssldump                 wifi-honey
 burpsuite              dumpzilla               intersect               ncrack                  safecopy                sslh                    wifitap
 cabextract             doona                   intrace                 ncurses-hexedit         readpst                 sslscan                 wifite
 cadaver                eapmd5pass              inviteflood             netdiscover             reaver                  sslsniff                windows-binaries
 casefile               edb-debugger            iodine                  netmask                 rebind                  sslstrip                wireshark
 cdpsnarf               enum4linux              irpas                   netsed                  recon-ng                sslsplit                wol-e
 cewl                   enumiax                 jad                     netsniff-ng             recordmydesktop         sslyze                  wordlists
 cgpt                   ethtool                 javasnoop               netwag                  recoverjpeg             statsprocessor          wpscan
 chirp                  ettercap-graphical      jboss-autopwn           nfspy                   redfang                 stunnel4                wvdial
 chkrootkit             ewf-tools               john                    ngrep                   redsocks                suckless-tools          xpdf
 chntpw                 exiv2                   johnny                  nikto                   reglookup               sucrack                 xprobe
 cisco-auditing-tool    exploitdb               joomscan                nipper-ng               regripper               swaks                   xspy
 cisco-global-exploiter extundelete             jsql                    nishang                 responder               t50                     xsser
 cisco-ocs              fcrackzip               keepnote                nmap                    rifiuti                 tcpflow                 xtightvncviewer
 cisco-torch            fern-wifi-cracker       keimpx                  ohrwurm                 rifiuti2                tcpick                  yersinia
 clang                  fierce                  killerbee               ollydbg                 rsmangler               tcpreplay               zaproxy
 clusterd               fiked                   king-phisher            onesixtyone             rtpbreak                termineter              zenmap
 cmospwd                fimap                   kismet                  ophcrack                rtpflood                tftpd32                 zim
 commix                 findmyhash              laudanum                ophcrack-cli            rtpinsertsound          thc-ipv6
 copy-router-config     flasm                   lbd                     oscanner                rtpmixsound             thc-pptp-bruter
 cowpatty               foremost                leafpad                 p0f                     sakis3g                 thc-ssl-dos
 creddump               fping                   libfindrtp              pack                    samdump2                theharvester
root@kali:~# 
```

Figure 2: Kali Linux Gnome full build

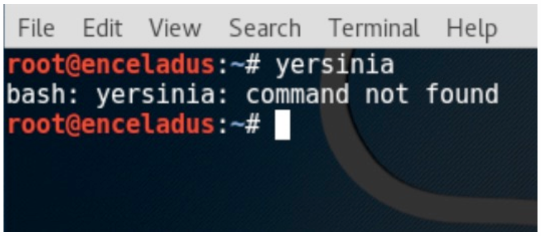
The command "apt-cache show kali-linux-full |grep Depends | tr ',' '\n' |column" will always show all of the packages in the kali-linux-full package. To test whether a particular package is on the OS, run the name of the package in the proper location. For example, running the command "yersinia" outputs the version number if the application was installed and outputs "bash: yersinia: command not found" if it is not available.



```
root@enceladus:~# yersinia
GNU yersinia 0.7.3
Try 'yersinia -h' to display the help.


MOTD: Magic people, voodoo people!
root@enceladus:~# 
```

Figure 3: Yersinia installed

Microsoft Office User



```
root@enceladus:~# yersinia
bash: yersinia: command not found
root@enceladus:~# 
```

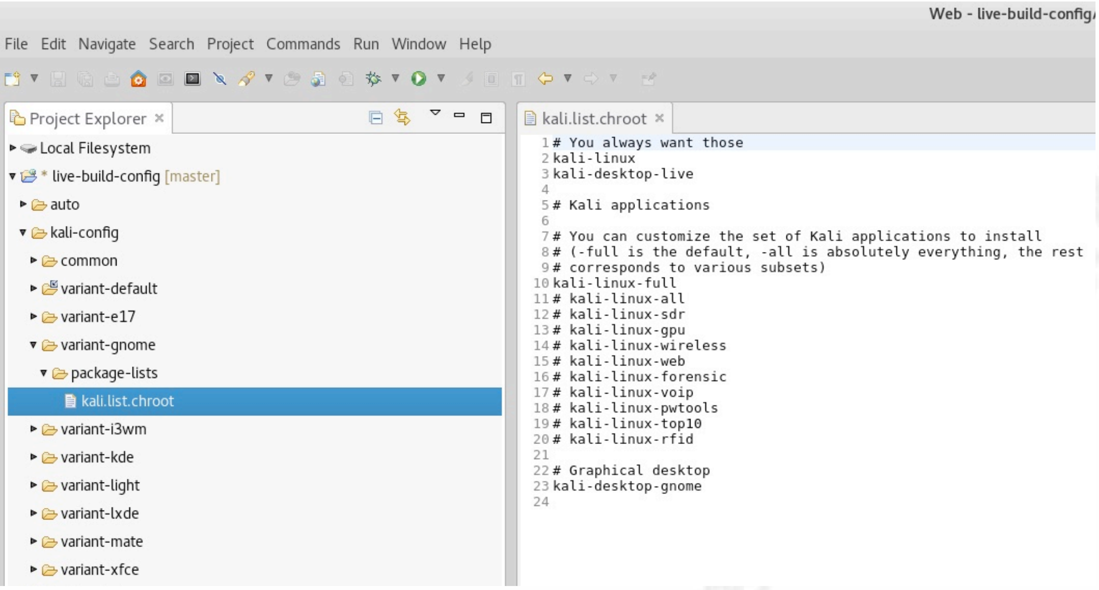
Figure 4: Yersinia not installed

Any Debian package can be included or removed to the core build illustrated on the first iteration of the VTM. However, it is recommended to avoid changing Offensive Security's code to assist with longevity and stability. Offensive Security is constantly updating the Kali Linux distro. Every new iteration of Kali Linux has a reduced chance of breaking the custom ISO image if minimal code changes occur to Kali's code base. Further customization is encouraged after the build is complete, as demonstrated in Section 2.3, Engineering.

Offensive Security groups packages into categories that can be used to further customize a Kali ISO image in an organized fashion. These categories include:

kali-linux-all,
kali-linux-full,
kali-linux-sdr,
kali-linux-gpu,
kali-linux-wireless,
kali-linux-web,
kali-linux-forensic,
kali-linux-voip,
kali-linux-pwtools,
kali-linux-top10, and
kali-linux-rfid.

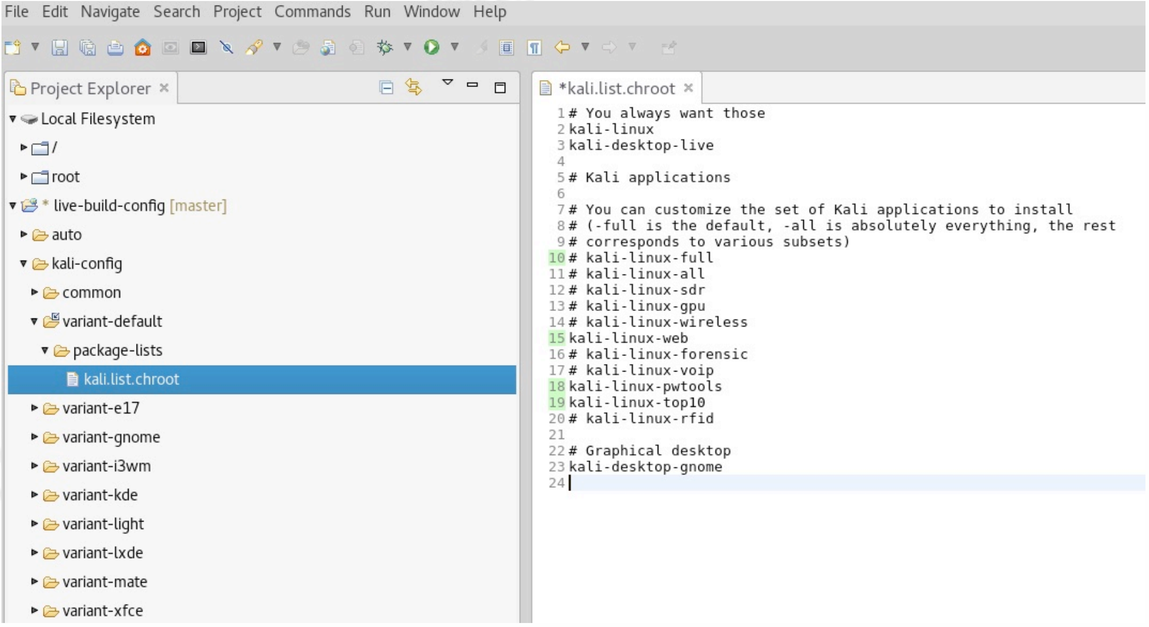
The following screenshot depicts the default configuration for a Kali Linux Gnome with the file "kali.list.chroot".



```
# You always want those
kali-linux
kali-desktop-live

# Kali applications

# You can customize the set of Kali applications to install
# (-full is the default, -all is absolutely everything, the rest
# corresponds to various subsets)
kali-linux-full
# kali-linux-all
# kali-linux-sdr
# kali-linux-gpu
# kali-linux-wireless
# kali-linux-web
# kali-linux-forensic
# kali-linux-voip
# kali-linux-pwtools
# kali-linux-top10
# kali-linux-rfid

# Graphical desktop
kali-desktop-gnome

```

Figure 5: File "kali-list.shroot" with default settings

To meet project specifications, "kali-linux-full" is commented out with a "#" symbol and "kali-linux-web", "kali-linux-pwtools", and "kali-linux-top10" are added by removing the "#" symbol in the file "/kali-config/variant-default/kali.list.chroot".



```
# You always want those
kali-linux
kali-desktop-live

# Kali applications

# You can customize the set of Kali applications to install
# (-full is the default, -all is absolutely everything, the rest
# corresponds to various subsets)
# kali-linux-full
# kali-linux-all
# kali-linux-sdr
# kali-linux-gpu
# kali-linux-wireless
kali-linux-web
# kali-linux-forensic
# kali-linux-voip
kali-linux-pwtools
kali-linux-top10
# kali-linux-rfid

# Graphical desktop
kali-desktop-gnome

```

Figure 6: File "kali-list.shroot" with modified settings

Nessus is a popular tool for vulnerability, configuration, and compliance assessments. Tenable offers a free version of Nessus for non-commercial use called Nessus Home (Tenable Network Security, 2016). The custom ISO will run as a live OS; thus, Nessus offline plug-ins will download directly from Tenable as shown in "setup.sh" in Section 3.2, Automation.

The Bro Network Security Monitor is a powerful, open-source intrusion detection system (IDS). Bro aids penetration testers by providing comprehensive network traffic analysis. Observing routine network traffic within a client environment provides numerous benefits including masquerading hidden malicious traffic, identifying weak or lacking encryption, and alerting on known misconfigurations (The Bro Project, 2016).

Burp Suite Professional is a necessary edition for any penetration tester assessing web applications. The free edition of Burp Suite comes with Kali Linux, but part of having a custom ISO image is having all licenses available for offline installation. Burp Suite Professional provides many advancements over the free edition including complex scanning, automated probes, and full target analysis (PortSwigger, 2016).

The following packages available through Kali repositories are available in the custom ISO image to meet project specifications:

brasero libreoffice keepass2 vsftpd filezilla xdotool freeipmi remmina virtualbox leafpad	remmina-plugin-gnome remmina-plugin-rdp remmina-plugin-vnc remmina-plugin-xdmcp chromium pepperflashplugin-nonfree icedtea-plugin flashplugin-nonfree virtualbox-dkms virtualbox-guest-additions-iso

2.3 Engineering

The engineering phase concentrates on taking applications from the design phase and utilizing Debian Live Systems to incorporate them at the proper time and location. The Debian Live Systems manual is currently up to date for the Debian 8.0 "Jessie" release (Debian.org, 2015). Because the ISO image's purpose is for penetration testing, it is important to engineer a solution to ensure the OS starts in passive mode without sending any packets on the client's network.

Debian Live-Build is a collection of scripts for creating custom live systems within Debian Live Systems. Debian Live-Build consists of three major commands that execute and process hundreds of minor commands. The three major commands include lb config, lb build, and lb clean. The "lb config" command creates the "/config/" directory, making up the initial skeleton configuration tree for the development environment. The "lb build" command reads the configuration files in the "/config/" directory and runs Live-Build's minor commands to create the ISO image. The "lb clean" command removes the chroot, binary, and source stages, allowing the "lb build" command to run multiple iterations without deleting the entire directory. The "lb clean" command does not clear the cache which assists in speeding up ISO creation (Debian.org, 2015).

Offensive Security scripted the "lb config" and "lb clean" commands in the files "/live-build-config/auto/config" and "/live-build-config/auto/clean", respectively. The heart of the engineering phase consists of placing the required files in the proper folders while changing Offensive Security's code as little as possible.

Live-Build installs all of the APT packages in the "/live-build-config/variant-default/package-lists/" directory at the proper time. For the custom ISO, create a file titled "custom.list.chroot" and include all of the APT packages required (Debian.org, 2015).

Live-Build installs all custom packages from the "config/packages.chroot/" directory after "lb config" is executed. Because Offensive Security scripted "lb config" and the custom build includes automation and version control, a more efficient process to install custom packages is to pass them directly through the new file system. To pass files through to the file system, Debian Live Systems established the "/live-build-config/includes.chroot/", "/live-build-config/includes.binary/", and "/live-build-config/includes.installer/" directories.

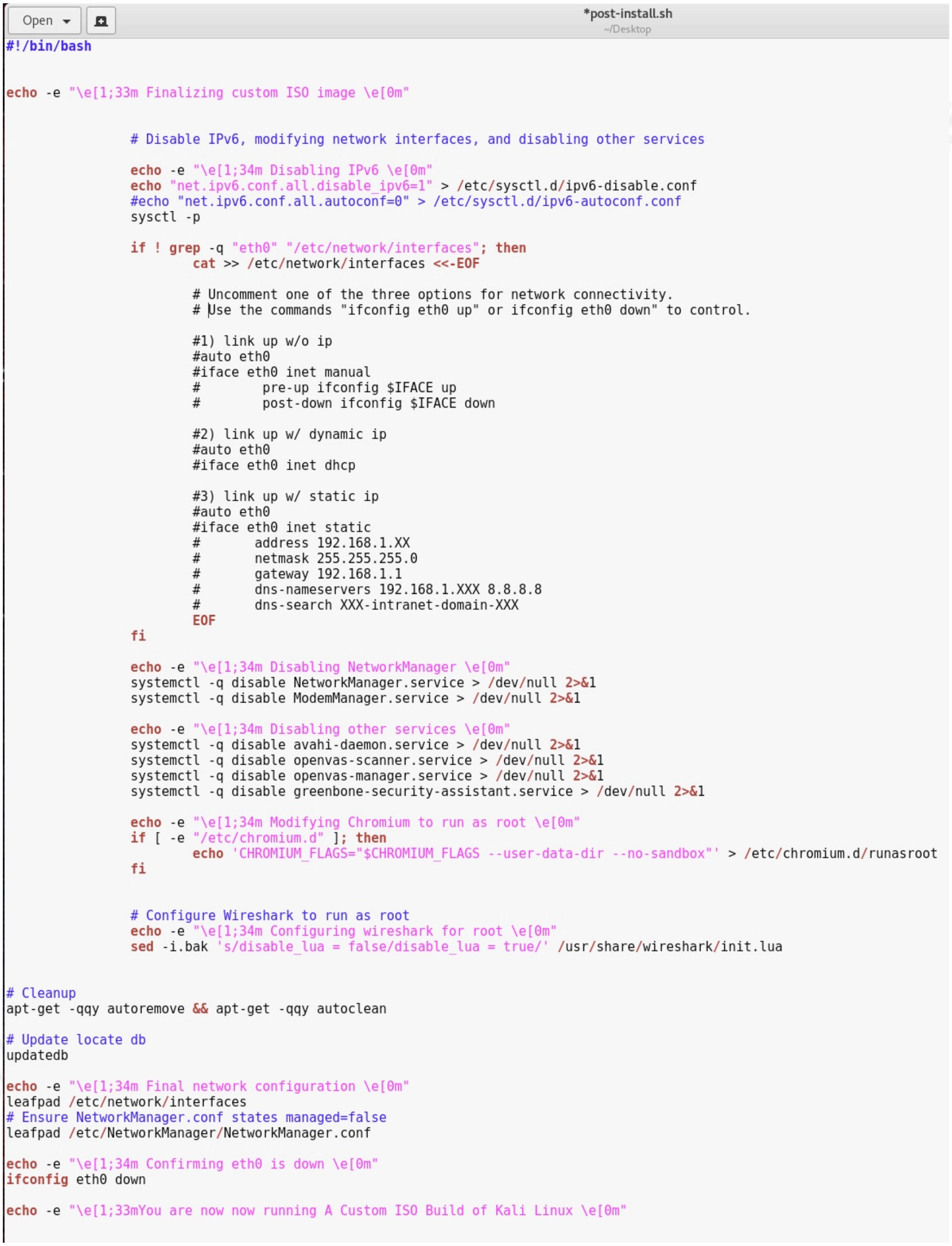
The custom build disables user accounts. Therefore, to pass a file directly in the root directory, use the "/live-build-config/kali-config/common/includes.chroot/root/" directory. Depending on specific needs, a common user account can be installed using the file "/live-build-config/includes.installer/preseed.cfg" and changing line 35 to "true" (Debian.org, 2015). Downloading Nessus with offline plug-ins, Burp Suite, and the Bro Network Security Monitor to the "/live-build-config/kali-config/common/includes.chroot/root/Additional-Apps/" directory, pipes the files through the installation. Due to licensing information, the URLs and filenames for the network paths to Nessus, Nessus offline plug-ins, Bro, and Burpsuite are missing from the following examples. The following commands create the proper directories and organize the aforementioned files in the correct location.

```
mkdir /root/Development/live-build-config/kali-
config/common/includes.chroot/root/Additional-Apps/

wget -O /root/Development/live-build-config/kali-
config/common/includes.chroot/root/Additional-Apps/Nessus6.6.deb
"Network or intranet path to Nessus"

wget -O /root/Development/live-build-config/kali-
config/common/includes.chroot/root/Additional-Apps/Nessus-Offline-
plugins.tar.gz "Network path to Nessus Plug-ins"

wget -O /root/Development/live-build-config/kali-
config/common/includes.chroot/root/Additional-Apps/Bro-2.4.1.tar.gz
"Network or intranet path to Bro"

wget -O /root/Development/live-build-config/kali-
config/common/includes.chroot/root/Additional-Apps/BurpSuite1.7.04.jar
" Network or intranet path to BurpSuite"
```

The final step in the engineering phase is to ensure the OS starts in passive mode without sending any packets on the client's network. Kali Linux is a complex OS with numerous applications and services requiring network connectivity. To verify that the custom ISO starts off in a quiet state, the goal is to disable most network services and configure the network interfaces in passive mode. One solution is to utilize Debian Live Systems to inject the specified network settings into the Custom ISO during creation.

Using Live Systems is not recommended, because the more code injected into Offensive Security's base code, the greater the risk of errors during update cycles.

The preferred method is to execute a premade script, "post-install.sh", piped directly into the ISO image during the build. The script opens the application Leafpad and forces the user to choose a network configuration setup. As long as the user does not plug in an Ethernet cable before running the script, no packets will transmit over the client's physical network. The following screenshot shows an example of "post-install.sh" that will get placed into the "/live-build-config/kali-config/common/includes.chroot/root/Additional-Apps/" directory.



```
#!/bin/bash


echo -e "\e[1;33m Finalizing custom ISO image \e[0m"


                # Disable IPv6, modifying network interfaces, and disabling other services

                echo -e "\e[1;34m Disabling IPv6 \e[0m"
                echo "net.ipv6.conf.all.disable_ipv6=1" > /etc/sysctl.d/ipv6-disable.conf
                #echo "net.ipv6.conf.all.autoconf=0" > /etc/sysctl.d/ipv6-autoconf.conf
                sysctl -p

                if ! grep -q "eth0" "/etc/network/interfaces"; then
                        cat >> /etc/network/interfaces <<-EOF

                        # Uncomment one of the three options for network connectivity.
                        # Use the commands "ifconfig eth0 up" or ifconfig eth0 down" to control.

                        #1) link up w/o ip
                        #auto eth0
                        #iface eth0 inet manual
                        #        pre-up ifconfig $IFACE up
                        #        post-down ifconfig $IFACE down

                        #2) link up w/ dynamic ip
                        #auto eth0
                        #iface eth0 inet dhcp

                        #3) link up w/ static ip
                        #auto eth0
                        #iface eth0 inet static
                        #       address 192.168.1.XX
                        #       netmask 255.255.255.0
                        #       gateway 192.168.1.1
                        #       dns-nameservers 192.168.1.XXX 8.8.8.8
                        #       dns-search XXX-intranet-domain-XXX
                        EOF
                fi

                echo -e "\e[1;34m Disabling NetworkManager \e[0m"
                systemctl -q disable NetworkManager.service > /dev/null 2>&1
                systemctl -q disable ModemManager.service > /dev/null 2>&1

                echo -e "\e[1;34m Disabling other services \e[0m"
                systemctl -q disable avahi-daemon.service > /dev/null 2>&1
                systemctl -q disable openvas-scanner.service > /dev/null 2>&1
                systemctl -q disable openvas-manager.service > /dev/null 2>&1
                systemctl -q disable greenbone-security-assistant.service > /dev/null 2>&1

                echo -e "\e[1;34m Modifying Chromium to run as root \e[0m"
                if [ -e "/etc/chromium.d" ]; then
                        echo 'CHROMIUM_FLAGS="$CHROMIUM_FLAGS --user-data-dir --no-sandbox"' > /etc/chromium.d/runasroot
                fi


                # Configure Wireshark to run as root
                echo -e "\e[1;34m Configuring wireshark for root \e[0m"
                sed -i.bak 's/disable_lua = false/disable_lua = true/' /usr/share/wireshark/init.lua


# Cleanup
apt-get -qqy autoremove && apt-get -qqy autoclean

# Update locate db
updatedb

echo -e "\e[1;34m Final network configuration \e[0m"
leafpad /etc/network/interfaces
# Ensure NetworkManager.conf states managed=false
leafpad /etc/NetworkManager/NetworkManager.conf

echo -e "\e[1;34m Confirming eth0 is down \e[0m"
ifconfig eth0 down

echo -e "\e[1;33mYou are now now running A Custom ISO Build of Kali Linux \e[0m"
```

Figure 7: File “post-install.sh”

Microsoft Office User

2.4 Troubleshooting

The troubleshooting phase tests the OS to ensure stability and is critical to customize an ISO image for use in a live state. Emphasizing repetition and thoroughness is essential for this phase.

Anytime a package is added to the basic Kali Linux custom ISO image, a meticulous test must verify overall stability. Testing requires taking a fully functional ISO file and installing it on the virtualization platform, VMware Fusion. All applications and network configurations should perform the same compared to the initial Kali Linux version downloaded from Kali Linux in Section 2.1, Planning. If an error occurs and can be duplicated in the original build downloaded from Offensive Security, the best solution is to report the issue to Offensive Security. Any fixed OS errors from Offensive Security's standpoint will not carry on to the latest release of Kali Linux's update cycle. The only fixable errors are those caused while customizing the ISO image.

Because of limitations with virtualization software, there may be errors running the ISO image in a live state in the virtual environment. To fully test the custom ISO image, always save the ISO image as a live-build (bootable image) on a USB drive or a DVD and verify the live-build on a laptop.

Because errors may occur during the troubleshooting phase, it is recommended to build the ISO image with the option "--verbose" and copy and paste any error messages into a search engine, such as Google. Numerous websites including StackExchange provide detailed mediations for common issues related to coding ISO images or programming in general.

2.5 Deploying

The deployment phase validates the custom ISO image by using real-world scenarios. The virtual environment significantly reduces development time and troubleshooting but does not fully test the ISO image during operations. To deploy this custom ISO image, use a live bootable USB drive or DVD. In extremely sensitive environments, it is recommended to use a DVD for security and convenience. Some networking environments explicitly deny USB drives and will confiscate any digital media. DVDs are very inexpensive and can easily be handed over to the organization after a test.

After inserting a burnable DVD-R that is recognized by OS X 10.11 Finder, run the command found in the man pages for hdiutil as follows:

```
hdiutil burn ~/Path/For/VTM.iso
```

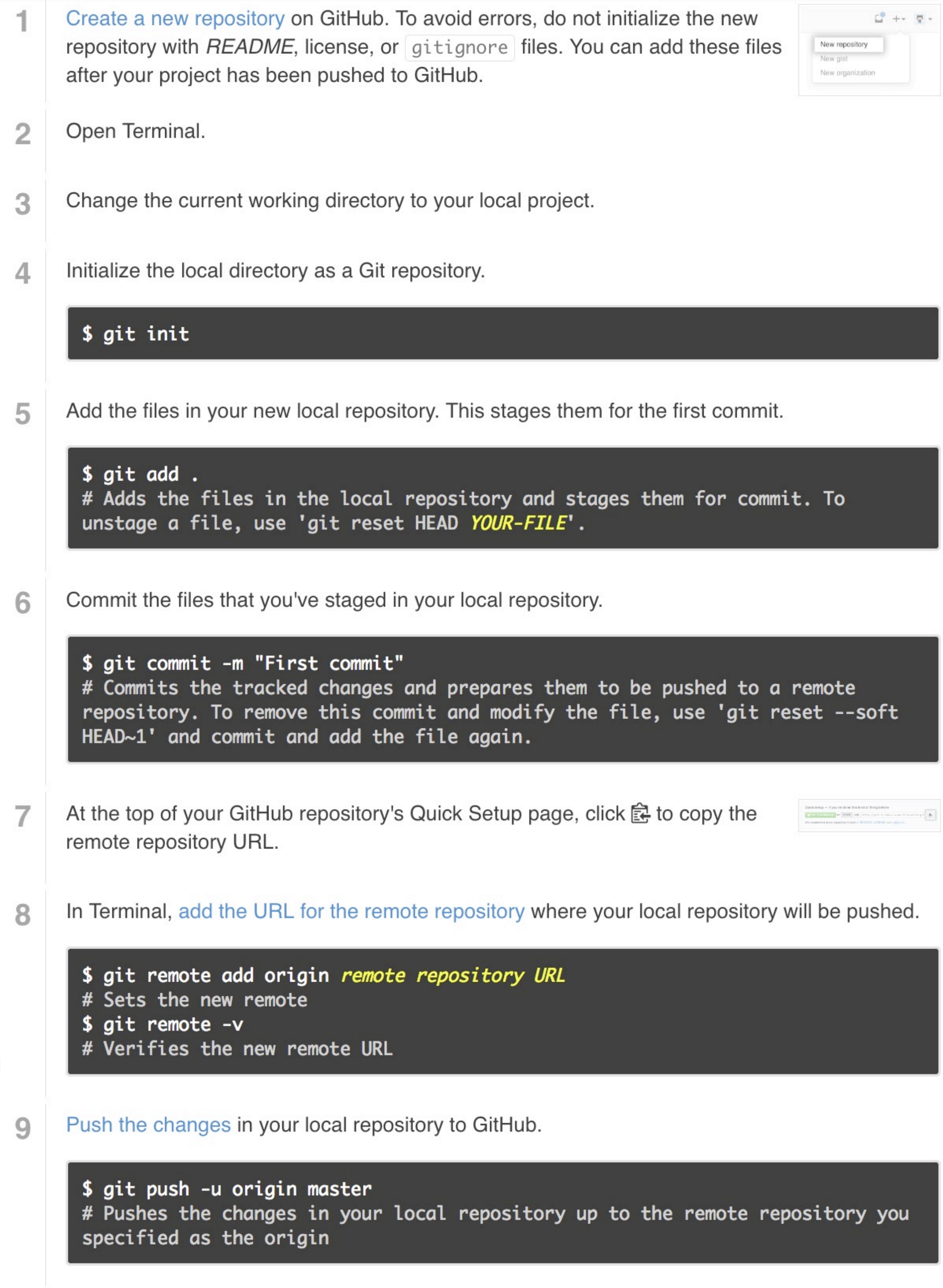
The “hdiutil” command is extremely versatile, and lots of additional options are available. Successfully running this command creates a live bootable image of the custom ISO file on the inserted DVD-R. Next, set up a laptop for a real-world penetration testing scenario that does not have a hard drive but has an optical drive that can read DVDs. During the deployment exercises, it is essential to keep the RJ-45 Ethernet port unplugged until after installing the post-installation scripts, as illustrated in Section 3.2, Automation.

3.0 Version Control and Automation

3.1 Version Control

Development environments are created to change an application’s code. However, any change may disrupt custom builds or automations. The goal of the versioning phase is to develop a system to track all changes, allow collaboration, and increase the stability of the code base. The more concise custom additions are to the ISO image, the less likely changes from update cycles will break the custom build.

First, set up a versioning environment. Git is an open-source version-control system used to version text in a distributed manner (Git, 2016). Offensive Security uses a Git repository to allow users to clone their build illustrated in Section 2.1, Planning. Multiple interpretations of Git are available for free, including GitLab and GitHub. Turnkey Linux provides a fully functioning VM that contains GitLab hosted on a Ruby on Rails website. GitLab provides code management from Git but with added control by hosting it on a private, dedicated server under the user’s control (Turnkey Linux, 2016). The fully functional, ready-to-go GitLab VM is available from https://www.turnkeylinux.org/gitlab. Detailed instructions for setting up a new Git project on GitHub or GitLab can be found at https://help.github.com/articles/create-a-repo/ and http://docs.gitlab.com/ce/gitlab-basics/create-project.html, respectively. Whether using GitLab on a dedicated server or GitHub on a public server, the commands for setting up a repository are the same. The following screenshot provided from GitHub documents creating a new repository (GitHub, 2016):



1 Create a new repository on GitHub. To avoid errors, do not initialize the new repository with *README*, license, or `gitignore` files. You can add these files after your project has been pushed to GitHub.

2 Open Terminal.

3 Change the current working directory to your local project.

4 Initialize the local directory as a Git repository.

```
$ git init
```

5 Add the files in your new local repository. This stages them for the first commit.

```
$ git add .
# Adds the files in the local repository and stages them for commit. To
unstage a file, use 'git reset HEAD YOUR-FILE'.
```

6 Commit the files that you've staged in your local repository.

```
$ git commit -m "First commit"
# Commits the tracked changes and prepares them to be pushed to a remote
repository. To remove this commit and modify the file, use 'git reset --soft
HEAD~1' and commit and add the file again.
```

7 At the top of your GitHub repository's Quick Setup page, click 📋 to copy the remote repository URL.

8 In Terminal, add the URL for the remote repository where your local repository will be pushed.

```
$ git remote add origin remote repository URL
# Sets the new remote
$ git remote -v
# Verifies the new remote URL
```

9 Push the changes in your local repository to GitHub.

```
$ git push -u origin master
# Pushes the changes in your local repository up to the remote repository you
specified as the origin
```

Figure 8: Creating a new Git repository on GitHub (GitHub, 2016)

Microsoft Office User

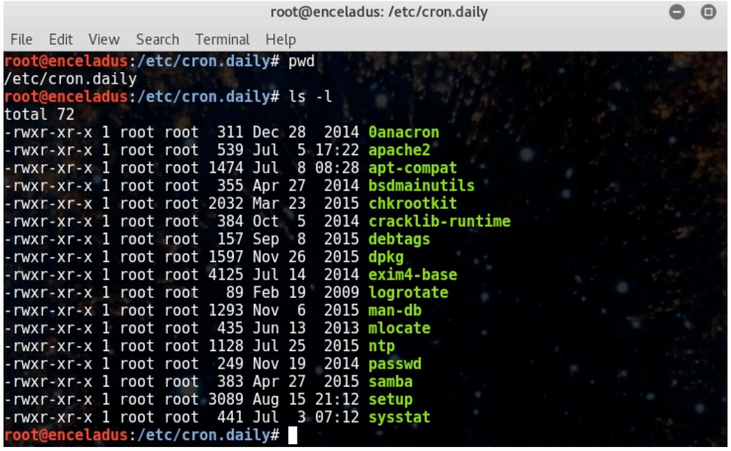
3.2 Automation

Most repetitive processes can be automated to increase efficiency and productivity. The goal of the automation phase is to create a script that incorporates all of the previous customizations and outputs a fully customized ISO image on a daily basis. Having a regular ISO image assists with redundancy and endurance. When creating an automated custom ISO builder, it's important to start in a stable development environment that maintains consistent uptime.

From KDVM, placing the executable script "setup.sh" in the /etc/cron.daily/ directory automates all of the customizations above and outputs a live bootable ISO image on the Desktop directory. After copying the file to /etc/cron.daily/, it is imperative to remove the file extension ".sh" to "setup" or the OS will ignore execution. It is recommended to maintain at least ten versions of the custom ISO image at all times to ensure the highest possible uptime. The "setup.sh" script renames each iteration with the current date to aid with redundancy and administration. To execute a script within the KDVM environment you may need to change the permissions of the file using the following command:

```
chmod +x setup.sh
```

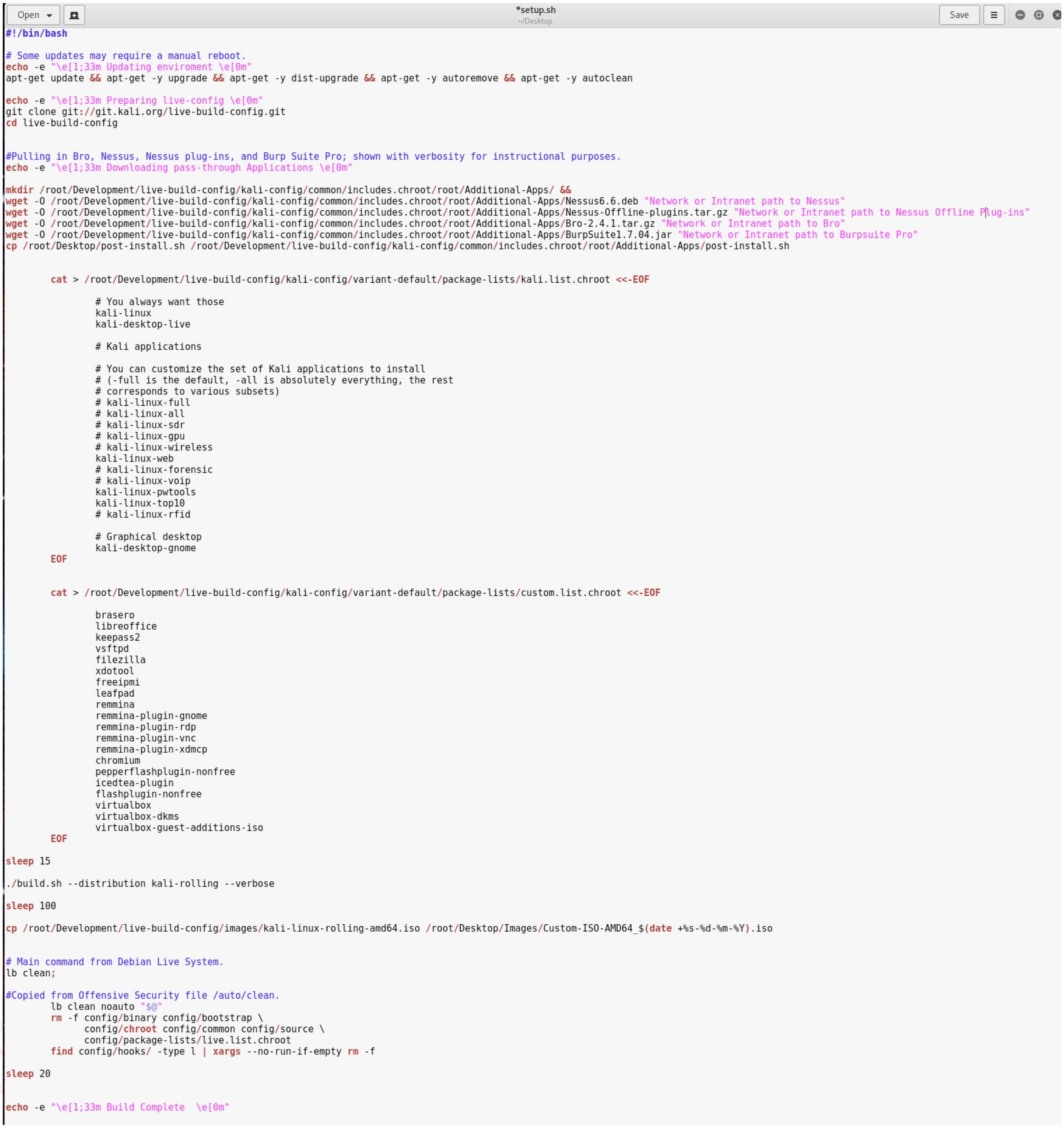
The following screen shot displays the /etc/cron.daily/ directory with the executable file "setup". The "setup" script does not include the ".sh" extension and matches the other /etc/cron.daily/ file permissions.


root@enceladus: /etc/cron.daily
File Edit View Search Terminal Help

```
root@enceladus:/etc/cron.daily# pwd
/etc/cron.daily
root@enceladus:/etc/cron.daily# ls -l
total 72
-rwxr-xr-x 1 root root  311 Dec 28  2014 0anacron
-rwxr-xr-x 1 root root  539 Jul  5 17:22 apache2
-rwxr-xr-x 1 root root 1474 Jul  8 08:28 apt-compat
-rwxr-xr-x 1 root root  355 Apr 27  2014 bsdmainutils
-rwxr-xr-x 1 root root 2032 Mar 23  2015 chkrootkit
-rwxr-xr-x 1 root root  384 Oct  5  2014 cracklib-runtime
-rwxr-xr-x 1 root root  157 Sep  8  2015 debtags
-rwxr-xr-x 1 root root 1597 Nov 26  2015 dpkg
-rwxr-xr-x 1 root root 4125 Jul 14  2014 exim4-base
-rwxr-xr-x 1 root root   89 Feb 19  2009 logrotate
-rwxr-xr-x 1 root root 1293 Nov  6  2015 man-db
-rwxr-xr-x 1 root root  435 Jun 13  2013 mlocate
-rwxr-xr-x 1 root root 1128 Jul 25  2015 ntp
-rwxr-xr-x 1 root root  249 Nov 19  2014 passwd
-rwxr-xr-x 1 root root  383 Apr 27  2015 samba
-rwxr-xr-x 1 root root 3089 Aug 15 21:12 setup
-rwxr-xr-x 1 root root  441 Jul  3 07:12 sysstat
root@enceladus:/etc/cron.daily# 
```

Figure 9: Directory /etc/cron.daily/ with setup script

Microsoft Office User



```
#!/bin/bash

# Some updates may require a manual reboot.
echo -e "\e[1;33m Updating enviroment \e[0m"
apt-get update && apt-get -y upgrade && apt-get -y dist-upgrade && apt-get -y autoremove && apt-get -y autoclean

echo -e "\e[1;33m Preparing live-config \e[0m"
git clone git://git.kali.org/live-build-config.git
cd live-build-config


#Pulling in Bro, Nessus, Nessus plug-ins, and Burp Suite Pro; shown with verbosity for instructional purposes.
echo -e "\e[1;33m Downloading pass-through Applications \e[0m"

mkdir /root/Development/live-build-config/kali-config/common/includes.chroot/root/Additional-Apps/ &&
wget -O /root/Development/live-build-config/kali-config/common/includes.chroot/root/Additional-Apps/Nessus6.6.deb "Network or Intranet path to Nessus"
wget -O /root/Development/live-build-config/kali-config/common/includes.chroot/root/Additional-Apps/Nessus-Offline-plugins.tar.gz "Network or Intranet path to Nessus Offline Plug-ins"
wget -O /root/Development/live-build-config/kali-config/common/includes.chroot/root/Additional-Apps/Bro-2.4.1.tar.gz "Network or Intranet path to Bro"
wget -O /root/Development/live-build-config/kali-config/common/includes.chroot/root/Additional-Apps/BurpSuite1.7.04.jar "Network or Intranet path to Burpsuite Pro"
cp /root/Desktop/post-install.sh /root/Development/live-build-config/kali-config/common/includes.chroot/root/Additional-Apps/post-install.sh


        cat > /root/Development/live-build-config/kali-config/variant-default/package-lists/kali.list.chroot <<-EOF

                # You always want those
                kali-linux
                kali-desktop-live

                # Kali applications

                # You can customize the set of Kali applications to install
                # (-full is the default, -all is absolutely everything, the rest
                # corresponds to various subsets)
                # kali-linux-full
                # kali-linux-all
                # kali-linux-sdr
                # kali-linux-gpu
                # kali-linux-wireless
                kali-linux-web
                # kali-linux-forensic
                # kali-linux-voip
                kali-linux-pwtools
                kali-linux-top10
                # kali-linux-rfid

                # Graphical desktop
                kali-desktop-gnome
        EOF


        cat > /root/Development/live-build-config/kali-config/variant-default/package-lists/custom.list.chroot <<-EOF

                brasero
                libreoffice
                keepass2
                vsftpd
                filezilla
                xdotool
                freeipmi
                leafpad
                remmina
                remmina-plugin-gnome
                remmina-plugin-rdp
                remmina-plugin-vnc
                remmina-plugin-xdmcp
                chromium
                pepperflashplugin-nonfree
                icedtea-plugin
                flashplugin-nonfree
                virtualbox
                virtualbox-dkms
                virtualbox-guest-additions-iso
        EOF

sleep 15

./build.sh --distribution kali-rolling --verbose

sleep 100

cp /root/Development/live-build-config/images/kali-linux-rolling-amd64.iso /root/Desktop/Images/Custom-ISO-AMD64_$(date +%s-%d-%m-%Y).iso


# Main command from Debian Live System.
lb clean;

#Copied from Offensive Security file /auto/clean.
        lb clean noauto "$@"
        rm -f config/binary config/bootstrap \
              config/chroot config/common config/source \
              config/package-lists/live.list.chroot
        find config/hooks/ -type l | xargs --no-run-if-empty rm -f

sleep 20


echo -e "\e[1;33m Build Complete  \e[0m"
```

Figure 10: Automation script, "setup.sh"

4.0 Summary

Automating a custom ISO image allows users to have a disposable OS image with all of the tools and licenses necessary to conduct an independent penetration test. Virtually all aspects of an ISO image can be tailored to fit the specific needs of a security team or large organization. Outside of Kali Linux, multiple OS distributions are available for customization to fulfill specific requirements, including:

SecurityOnion (securityonion.net),
BlackArch (blackarch.org),
Tails (tails.boum.org),
Samurai Web Testing Framework (samurai.inguardians.com), and
BackBox (backbox.org).

If Kali Linux is the best choice for a customizable ISO image, it is important to take into account the varieties of images Offensive Security supports. Offensive Security's recent switch to a rolling environment may create additional difficulties for versioning a custom ISO image from a rolling-code base. Because of these possibilities, this project stressed changing Offensive Security's code as little as possible while striving to create a unique and efficient live bootable OS. Applications outside of Kali's repositories were piped through the ISO creation process and into specific directories inside the root directory.

It is important to note that Offensive Security still maintains older distributions of Kali Linux. These include different graphical interfaces, process architectures, Debian versions, and numerous other variations located on https://www.kali.org/downloads/. If the development environment runs on a frequent basis, it may be useful to setup a network mirror of Offensive Security's repositories. Offensive Security provides detailed instructions to setup a Kali mirror at http://docs.kali.org/community/kali-linux-mirrors.

This project created a custom ISO image specifically tailored for a closed environment. Wireshark recommends running Wireshark or TShark with limited user privileges to prevent serious vulnerabilities associated with the application (Wireshark, 2016). Further details regarding security issues for Wireshark and TShark are located at https://wiki.wireshark.org/Security.

References

McFarland, R. (2014). Wireless Security. In Information Security Basics: Fundamental Readings for INFOSEC including the CISSP, CISM, CCNA-SECURITY Certification Exams [Kindle Edition].

National Institute of Standards and Technology (NIST). (2013). Security and Privacy Controls for Federal Information Systems and Organizations. NIST Special Publication, 800-53(4). Retrieved from http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf

Federal Information Processing Standards (FIPS). (2006). Minimum Security Requirements for Federal Information and Information Systems. Federal Information Processing Standards (FIPS) Publications, 200. Retrieved from http://csrc.nist.gov/publications/fips/fips200/FIPS-200-final-march.pdf

Scott, T. (2016). Introduction. In Cybersecurity Information Gathering using Kali Linux.

Offensive Security. (2016a). What is Kali Linux? Retrieved from http://docs.kali.org/introduction/what-is-kali-linux.

ISO. (2015). ISO 9660:1988 Information processing -- Volume and file structure of CD-ROM for information interchange. Retrieved from http://www.iso.org/iso/home/store/catalogue_tc/ catalogue_detail.htm?csnumber=17505.

Library of Congress. (2012). ISO Disk Image File Format. Retrieved from http://www.digitalpreservation.gov/formats/fdd/fdd000348.shtml.

ECMA. (1997). Volume and File Structure for Write-Once and Rewritable Media using Non-Sequential Recording for Information Interchange [Standard ECMA-167] (3rd ed.). Retrieved from http://www.ecma.ch.

Offensive Security. (2016b). Kali Linux, Rolling Edition Released – 2016.1. Retrieved from https://www.kali.org/news/kali-linux-rolling-edition-2016-1/.

Sak, B. (2016). Wireless penetration testing fundamentals. In Mastering Kali Linux Wireless Pentesting (p. 35).

Boyde, J. (2014). Project Life Cycle. In A Down-to-Earth Guide to SDLC Project

Microsoft Office User

Management: Getting Your System/Software Development Life Cycle Project Successfully across the Line Using PMBOK Adaptively (2nd ed., p. 39).

Appcelerator, Inc. (2014). Aptana Studio 3. Retrieved from http://www.aptana.com/.

Debian. (2015). The Debian Administrator's Handbook. Retrieved from https://debian-handbook.info/browse/stable/sect.apt-cache.html.

Tenable Network Security. (2016). Download Nessus. Retrieved from http://www.tenable.com/products/nessus/select-your-operating-system.

The Bro Project. (2016). The Bro Network Security Monitor. Retrieved from https://www.bro.org/index.html.

PortSwigger. (2016). Burp Suite. Retrieved from https://portswigger.net/burp/.

Debian.org. (2015). Live Systems Manual. Retrieved from https://debian-live.alioth.debian.org/live-manual/stable/manual/html/live-manual.en.html.

Git. (2016). Git --everything-is-local. Retrieved from https://git-scm.com.

Turnkey Linux. (2016). GitLab - Self Hosted Git Management. Retrieved from https://www.turnkeylinux.org/gitlab.

GitHub. (2016, July 18). [Creating a new Git repository on GitHub]. Retrieved from https://help.github.com/articles/creating-a-new-repository/.

Wireshark. (2016). CaptureSetup / CapturePrivileges. Retrieved from https://wiki.wireshark.org/CaptureSetup/CapturePrivileges.

Microsoft Office User