

Edith Cowan University

Research Online

Australian Information Security Management
Conference

Conferences, Symposia and Campus Events

2018

Mitigating man-in-the-middle attacks on mobile devices by blocking insecure http traffic without using vpn

Kevin Chong
Edith Cowan University

Muhammad Imran Malik
Edith Cowan University

Peter Hannay

Follow this and additional works at: <https://ro.ecu.edu.au/ism>



Part of the [Information Security Commons](#)

DOI: [10.25958/5c526c2966688](https://doi.org/10.25958/5c526c2966688)

Chong, K., Malik, M.I., & Hannay, P. (2018). Mitigating man-in-the-middle attacks on mobile devices by blocking insecure http traffic without using vpn. In *proceedings of the 16th Australian Information Security Management Conference* (pp. 1-13). Perth, Australia: Edith Cowan University.

This Conference Proceeding is posted at Research Online.

<https://ro.ecu.edu.au/ism/224>

MITIGATING MAN-IN-THE-MIDDLE ATTACKS ON MOBILE DEVICES BY BLOCKING INSECURE HTTP TRAFFIC WITHOUT USING VPN

Kevin Chong¹, Muhammad Imran Malik¹, Peter Hannay^{1,2}

¹School of Science, Edith Cowan University, ²Asterisk Information Security, Perth, Australia
kkchong0@our.ecu.edu.au, muhammad.malik@ecu.edu.au, peter.hannay@asteriskinfosec.com.au

Abstract

Mobile devices are constantly connected to the Internet, making countless connections with remote services. Unfortunately, many of these connections are in cleartext, visible to third-parties while in transit. This is insecure and opens up the possibility for man-in-the-middle attacks. While there is little control over what kind of connection running apps can make, this paper presents a solution in blocking insecure HTTP packets from leaving the device. Specifically, the proposed solution works on the device, without the need to tunnel packets to a remote VPN server, and without special privileges such as root access. Speed tests were performed to quantify how much network speed is being impacted while filtering. To investigate how blocking HTTP traffic can affect day-to-day usage, common tasks were put to the tests, tasks such as browsing, searching, emailing, instant messaging, social networking, consuming streaming content, and gaming. The results from the tests are interesting, websites that do not support HTTPS were exposed, apps that do not fully support HTTPS were also being uncovered. One surprisingly, and arguably pleasant, side effect was discovered – the filtering solution blocks out advertisements in all of the games being tested, hence contributing to an improved gaming experience.

Keywords

Filtering, blocking, on-device, security, virtual network interface, mobile device, Android

INTRODUCTION

Mobile devices commonly connect to untrusted wireless access points, this is especially so while travelling. Connection to untrusted network presents a risk for man-in-the-middle attacks, making eavesdropping, content injection, and cookie stealing possible (Electronic Frontier Foundation, n.d.). However, man-in-the-middle attacks are possible only when the connection is not encrypted end to end, such as that provided by HTTP protocol. By denying HTTP requests, the risk of exposing the device to man-in-the-middle attacks will effectively be reduced.

Applying a blanket denial of all HTTP requests might not be as outrageous as it sounds. Internet giants and security experts have been advocating web security for a number of years now (“Encrypt All the Things”, n.d.; Google Developers, 2014; Schechter, 2016; Vyas & Dolanjski, 2017; Wagenseil, 2014). Leading browsers are starting to name and shame HTTP websites (Claburn, 2018; Rutherford, 2017; Schechter, 2018), and devising plans to deprecate non-secure HTTP (Barnes, 2015). Android and iOS are introducing plans to drop insecure connections (Conger, 2016; Thierer, 2017). Google is even ranking HTTPS sites higher than their HTTP counterparts (Bahajji & Illyes, 2014).

This paper looks at an on-device filtering solution to block out insecure HTTP requests only. The proposed solution being investigated will be confined to the Android operating system, and will not require the device to be rooted.

EXISTING VPN SOLUTIONS AND THEIR DRAWBACKS

VPN, or Virtual Private Network, is a common solution used to combat against man-in-the-middle attacks. VPN establishes an encrypted tunnel between the device and the VPN server, even though the untrusted network still sits in between, a man-in-the-middle attack is not possible since all traffic is now protected by encryption as it goes through the tunnel (Siciliano, 2017), as shown in Figure 1 below.

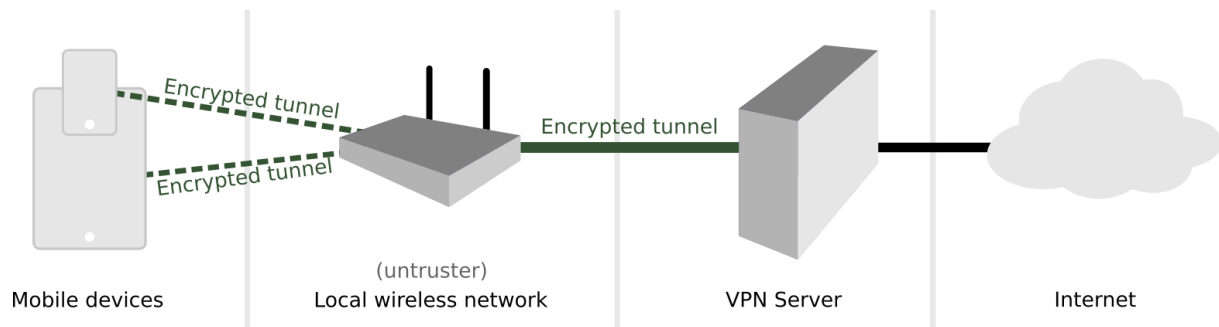


Figure 1: Secure encrypted tunnels established between devices and VPN server

There are however a number of drawbacks with VPN:

1. Requirement for VPN server – This adds an additional dependency on an external service. Additional dependency introduces another point of failure into the system.
2. Costs involved – There is VPN server setup, running, and maintenance costs involved. For those who choose to use third-party service, there is service fee involved. Affordability can be a barrier for some.
3. Setup is required on mobile devices – Depending on the type of VPN, the setup might not be trivial, and this could potentially put users off.
4. VPN service not guaranteed to be available at all times – With any other services, no VPN providers can guarantee an uptime of 100%. During the times when the VPN service is down, encryption will also be unavailable.
5. Network speed is affected – Since every network packet gets routed through the VPN server, network speed will be degraded if the VPN server, or required connections, are not performant.
6. Third-party VPN service cannot fully be trusted – There is the risk that man-in-the-middle attacks can be carried out by the VPN provider, or by attackers if in the event that the VPN server is compromised. The risk is not mitigated under such a scenario, it is simply shifted from the local network operator to the VPN provider.
7. VPN traffic can be blocked – VPN traffic is known to be blocked by some firewalls which are typically outside of user's control. It is also known to be blocked in certain countries such as China ("VPN blocking", 2018).
8. Traffic not encrypted end-to-end – The encrypted tunnel only extends from the device to the VPN server. Once traffic leaves the VPN server, unencrypted HTTP traffic will no longer be protected.

A related study investigating privacy leaks from mobile devices took a similar approach, the mobile device connects to the Internet via a VPN proxy sitting in between, where packets are being inspected (Jingjing, Ashwin, Martina, Arnaud, & David, 2016). Since this approach has a reliance on an external VPN proxy, it suffers from the same drawbacks as discussed above.

ON-DEVICE FILTERING SOLUTION

This paper presents a filtering solution which does not rely on an external VPN service, it is an on-device solution and does not require the Android device to be rooted. Note that because of the non-rooted requirement, running a local VPN server on the device is ruled out since raw sockets are only available with root privilege.

Fortunately, Android SDK provides a virtual network interface to app developers where all traffic gets routed through. The interface, which operates at the IP layer, provides the app with a file descriptor. Each read from the descriptor retrieves an outbound IP packet, and each write to the description injects an inbound IP packet.

This ability given to regular apps to intercept network traffic without root access is via the API class `VpnService` ("Android SDK `VpnService`", n.d.). A number of studies conducted previously which required on-device packet interception and inspection also used `VpnService` in their research (Ikram & Kaafar, 2017; Ozsoy, 2016; Shuba, 2016; Song & Hengartner, 2015; Trivedi & Das, 2017).

The approach here is to let the filter app examine every outgoing packet. If an outgoing packet is an insecure HTTP packet, the filter app will block it by not letting it go any further, otherwise, the filter app will make necessary connections with the packet's target server and route it to its rightful destination. On the reverse side, when the

filter app gets incoming traffic from these opened connections, it needs to route the traffic back into the virtual network interface.

How HTTP Traffic is Blocked

As illustrated in Figure 2, this is how HTTP traffic initiated from an app within the device is blocked by the filter app from ever leaving the device:

1. An app initiates an HTTP connection with a remote host at example.com.
2. Android routes HTTP packet to the virtual network interface provided by VpnService.
3. Filter app gets the packet from the virtual network interface.
4. Filter app examines the packet.
5. Since it is an HTTP packet, filter app does nothing further with it, essentially blocking it from leaving the device.

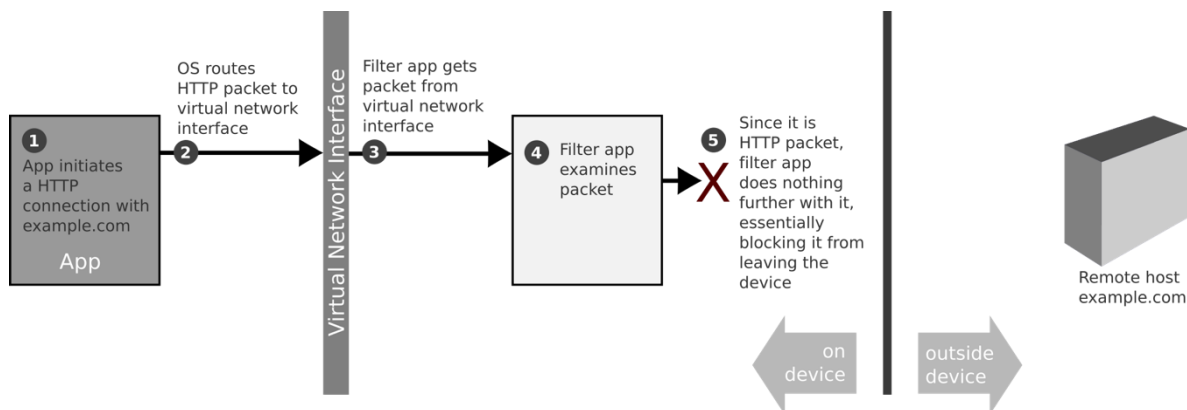


Figure 2: How HTTP Traffic is Blocked by the Filter App

How HTTPS Traffic is Allowed

The following Figure 3 illustrates how HTTPS traffic is not blocked:

1. An app initiates an HTTPS connection with a remote host at example.com.
2. Android routes HTTPS packet to the virtual network interface provided by VpnService.
3. Filter app gets the packet from the virtual network interface.
4. Filter app examines the packet.
5. Since it is not an HTTP packet, filter app establishes a connection with the remote host.
6. Once the connection is established, filter app routes the traffic to the remote host.
7. Filter app gets incoming traffic from the remote host via the connection.
8. Filter app routes incoming packets into the virtual network interface.
9. Android routes incoming packets to the app that initiated the HTTPS connection.

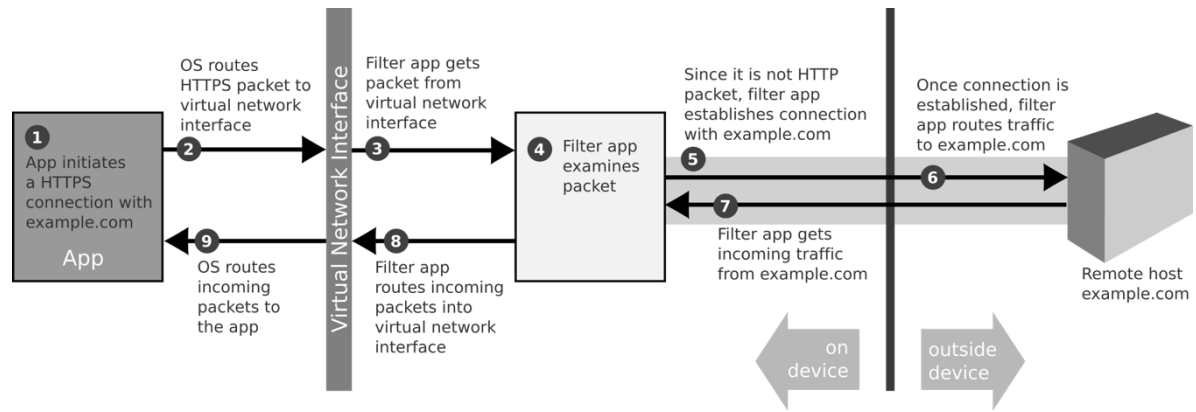


Figure 3: How HTTPS Traffic is Allowed by the Filter App

Implementation of the Filtering Solution

There is an open source Android app called NetGuard, which uses VpnService to filter traffic by IP address (Bokhorst, 2018). NetGuard, however, does not do exactly what is needed in this research. Since its source code is open, required modifications can be made to block all HTTP traffic.

One change was made in the source code for this study, it is found on line 301 in app/src/main/jni/nethuard/ip.c file, as shown in Figure 4. This line of change simply instructs the app to block the outbound IP packet if it is a TCP packet and its remote port is 80, which is the standard port for HTTP protocol. The source code is available at GitHub source code repository (Chong, 2018).

Note that the change introduced in ip.c file will unfortunately not block HTTP traffic connecting to other ports. Additionally, although very unlikely, this can incorrectly block non-HTTP traffic connecting to remote port 80. For the scope of this research, it is reasonable to assume that all HTTP traffic, and no other traffic, is connecting to remote port 80 from the device.

<pre> 290 291 // Check if allowed 292 int allowed = 0; 293 struct allowed *redirect = NULL; 294 if (protocol == IPPROTO_UDP && has_udp_session(args, pkt, payload)) 295 allowed = 1; // could be a lingering/blocked session 296 else if (protocol == IPPROTO_TCP && !syn) 297 allowed = 1; // assume existing session 298 else { 299 jobject objPacket = create_packet(300 args, version, protocol, flags, source, sport, dest, dport, "", uid, 0); 301 redirect = is_address_allowed(args, objPacket); 302 allowed = (redirect != NULL); 303 if (redirect != NULL && (*redirect->raddr == 0 redirect->rport == 0)) 304 redirect = NULL; 305 } 306 </pre> <p style="text-align: center;">Before</p>	<pre> // Check if allowed int allowed = 0; struct allowed *redirect = NULL; if (protocol == IPPROTO_UDP && has_udp_session(args, pkt, payload)) allowed = 1; // could be a lingering/blocked session else if (protocol == IPPROTO_TCP && !syn) allowed = 1; // assume existing session else { jobject objPacket = create_packet(args, version, protocol, flags, source, sport, dest, dport, "", uid, 0); redirect = (protocol == IPPROTO_TCP && dport == 80) ? NULL : is_address_allowed(args, objPacket); allowed = (redirect != NULL); if (redirect != NULL && (*redirect->raddr == 0 redirect->rport == 0)) redirect = NULL; } </pre> <p style="text-align: center;">After</p>
--	---

Figure 4: Change made to NetGuard source code in app/src/main/jni/nethuard/ip.c

Once the required change is made, the modified NetGuard can be built using Android Studio or the command line tools ("Android Studio", n.d.). Besides the minimum requirement of Android version 5.1, NetGuard is just a regular app with no special prerequisites. One setting needs to be configured within the app for it to filter HTTP traffic – which is to turn on the 'Filter traffic' switch on the 'Advanced options' screen as shown in Figure 5. The 'Advanced options' screen is accessed via three-dot menu (top right on NetGuard's main screen) > Settings > Advanced options.

To start filtering, the switch at the top left on NetGuard's main screen needs to be turned on, as shown in Figure 6. Once filtering is switched on, visiting an HTTP website is no longer possible (Figure 7) but visiting an HTTPS website is unaffected (Figure 8).

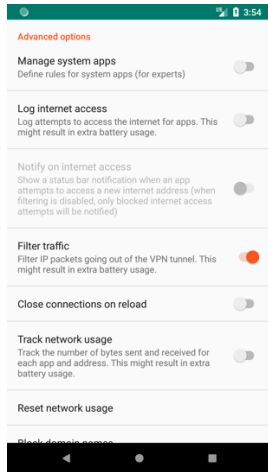


Figure 5: NetGuard configured to run in filter mode

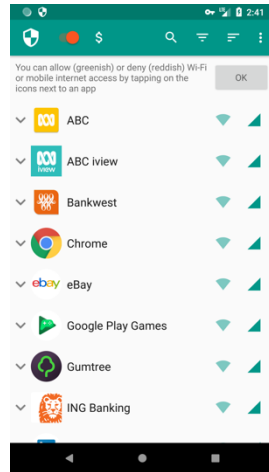


Figure 6: Turning on filtering (orange switch at the top of NetGuard's main screen). The icon which looks like a key on the system status bar indicates that filtering is turned on for the device.

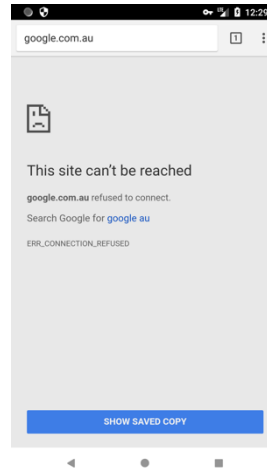


Figure 7: Insecure HTTP traffic is blocked

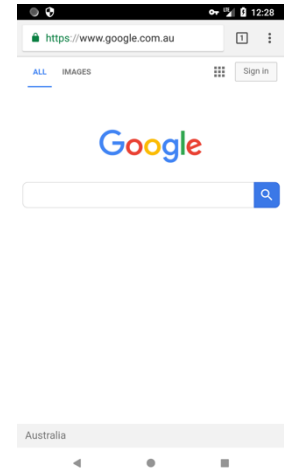


Figure 8: Secure HTTPS traffic is allowed

SPEED TESTS

Since each IP packet is being intercepted by NetGuard for examination, this can slow down the traffic flow. To measure how much the download speed is affected, some tests were conducted on the device using an app called Speedtest by Ookla ("Speedtest by Ookla", n.d.). The result of the tests is being plotted in Figure 9.

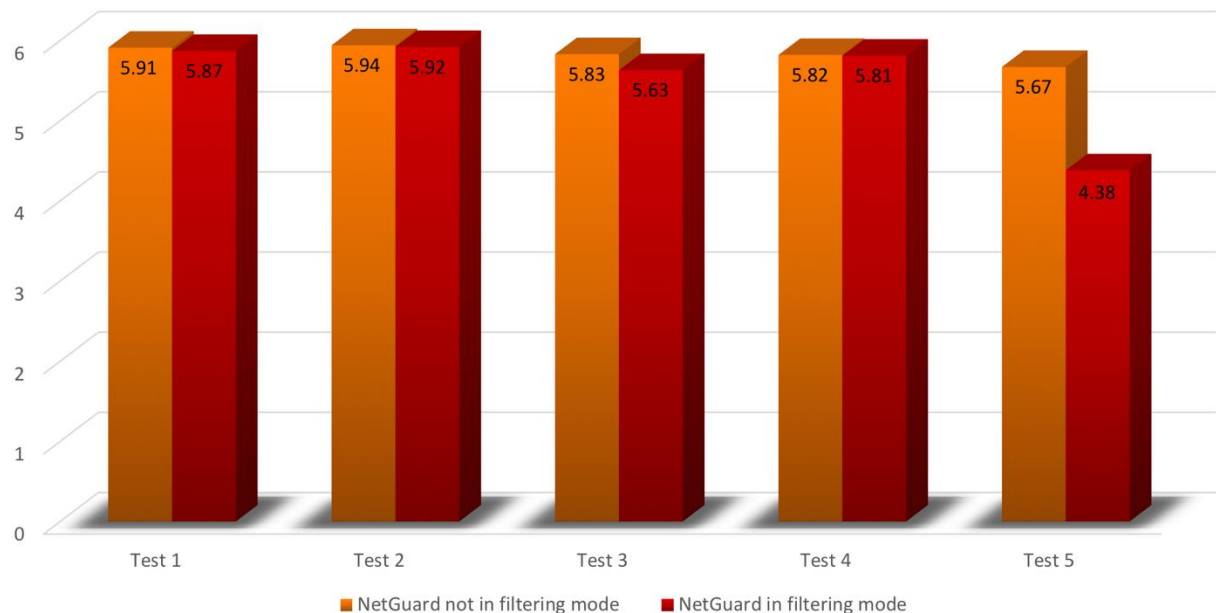


Figure 9: Download speeds in Mbps

The tests show that there is hardly any decrease in the download speed while the traffic is being filtered by NetGuard. Apart from Test 5, the download speed for when NetGuard is filtering is almost identical to the download speed for when NetGuard is not filtering, on average, it is just 5.47% slower. With this result, it is reasonable to conclude that the proposed filtering solution does not negatively impact the network speed of the device.

DAY-TO-DAY MOBILE USAGE TESTS











In the next set of tests, we want to find out if day-to-day mobile device usage is being affected by this filtering solution, and if so, how is it being impacted. Besides the telephony functions, such as making calls and texting via SMSs, a list of tasks commonly performed by mobile users in recent times (“REVEALED: Top uses of our smartphones - and calling doesn't even make the list”, 2017; “The most common usage of a mobile phone”, 2015) are being compiled and put under tests, these tasks are:

1. Browsing
2. Emailing
3. Searching
4. Social networking
5. Instant messaging
6. Watching and listening to streaming contents
7. Gaming
8. Reading news and checking weather
9. Navigation
10. Banking
11. Taking pictures
12. Using other common apps

Browsing

The goal of this test is to assess how blocking can affect the browsing experience. To do this, top 10 sites in Australia according to Alexa (“Top Sites in Australia”, 2018) were put to the test. Chrome browser app was used for this test, note that ‘https://’ had to be explicitly entered into the URL, otherwise the browser would default to HTTP protocol. Results of this test are tabulated below in Table 1.

Table 1: Results of Visiting Top 10 Sites in Australian Using HTTPS




Raking	Website	Result	Notes
1	 https://google.com.au	Passed	Works flawlessly.
2	 https://youtube.com	Passed	Works flawlessly.
3	 https://google.com	Passed	Works flawlessly.
4	 https://facebook.com	Passed	Works flawlessly.
5	 https://reddit.com	Passed	Works flawlessly.
6	 https://wikipedia.org	Passed	Works flawlessly.
7	 https://ebay.com.au https://www.ebay.com.au	Failed Passed	https://ebay.com.au redirects to http://ebay.com.au
8	 https://live.com	Passed	Works flawlessly.
9	 https://twitter.com	Passed	Works flawlessly.
10	 https://netflix.com	Passed	Works flawlessly.

All the above sites could be visited using HTTPS. One interesting note regarding eBay’s website, it failed for <https://ebay.com.au> but if the user typed in <https://www.ebay.com.au> instead, it would work fine. The concluding remark from this test is that visiting popular sites are not being impacted by the filtering solution, the one minor inconvenience is that this mandates the user to type ‘https://’ at the beginning of the website address, otherwise the browser would assume using HTTP protocol.

Emailing

The three most popular email apps were put to the test. Results of this test are tabulated below in Table 2.

Table 2: Results of Using Popular Email Apps




App	Result	Notes
 Gmail	Passed	Works flawlessly, including notifications on new email.
 Microsoft Outlook	Passed	Works flawlessly, including notifications on new email.
 Yahoo Mail	Passed	Works flawlessly, including notifications on new email.

All three apps work flawlessly in this test. No issues found with both sending and receiving emails, with or without attachments. Notifications also work as usual whenever a new email arrives.

Searching

The three most used search engines are Google, Bing, and Yahoo. These search engines were visited using the Chrome browser app, the results are tabulated in Table 3 below.



Table 3: Results of Using Popular Search Engines

Search Engine	URL	Result	Notes
 Google	https://google.com	Passed	Works flawlessly.
 Bing	https://bing.com	Passed	Works flawlessly.
 Yahoo	https://yahoo.com	Passed	Works flawlessly.

Performing searching using the above search engines work flawlessly, search results are returned as per normal. However, if a result's link is using HTTP protocol, clicking on that link will fail as expected. In such a case, the user will have to manually change 'http://' to 'https://' with the hope that the site has support for HTTPS.

Google and Bing have their own app, they were also put under test, with the results tabled below in Table 4.

Table 4: Results of Using Search Engine Apps





App	Result	Notes
 Google	Passed	Works flawlessly.
 Bing	Passed	Works flawlessly.

The experience of using these apps is similar to using the browser, search results were returned as per normal. As with using the browser, if a result's link is in HTTP protocol, clicking on that link will fail to load.

Social Networking

Some of the most popular social network apps were put to the test next, the results are shown in Table 5 below.

Table 5: Results of Using Popular Social Network Apps




App	Result	Notes
 Facebook	Passed	Works flawlessly, including notifications.
 Twitter	Passed	Works flawlessly, including notifications.
 Instagram	Passed	Works flawlessly, including notifications.
 LinkedIn	Passed	Works flawlessly, including notifications.

There are no issues using all these apps, they function normally in every aspect. Notifications also work fine. As expected, any outbound links in HTTP would be blocked by the filtering solution, and hence would fail to load.

Instant Messaging

Next, instant messaging apps were being tested, voice and video calling capabilities were also tested. The results are documented in Table 6.

Table 6: Results of Using Popular Instant Messaging Apps








App	Result	Notes
 Messenger	Passed	Works flawlessly. Receiving and sending text messages and photos work well, receiving and making voice and video calls also work well. All notifications are being received.
 Hangouts	Passed	Works flawlessly. Receiving and sending text messages and photos work well, receiving and making voice and video calls also work well. All notifications are being received.
 WhatsApp	Passed	Works flawlessly. Receiving and sending text messages and photos work well, receiving and making voice and video calls also work well. All notifications are being received.

There are no issues here, all of the above apps work flawlessly in every way.

Media Streaming

In this test, video and audio streaming apps were being investigated, the results are tabulated in Table 7 below.

Table 7: Results of Using Popular Media Streaming Apps



App	Result	Notes
 YouTube	Passed	Works flawlessly.
 Vimeo	Passed	Works flawlessly.
 Spotify	Passed	Works flawlessly.
 Google Play Music	Passed	Works flawlessly.
 Netflix	Passed	Works flawlessly.
 ABC iview	Failed	Textual and graphics contents are showing fine on the app, but none of the video contents can be played.
 SBS On Demand	Failed	Textual and graphics contents are showing fine on the app, but none of the video contents can be played.




Besides ABC iview and SBS On Demand apps, no issues were found while using all the other apps. All contents, be it textual or graphical, video or audio, all work flawlessly. ABC iview and SBS On Demand apps appear to be working fine initially since all textual and graphics contents are displaying correctly, but none of the videos can be played.

Gaming

In this test, some of the top games listed on Google Play Store were put to the test. The results are tabulated below in Table 8.

Table 8: Results of Playing Games

App	Result	Notes
 Helix Jump	Passed	Game itself is not affected. Advertisements which normally appear at the bottom of the screen are no longer being shown. Intermittent advertisement screens are also no longer being shown between levels.
 Flip the Gun	Passed	Game itself is not affected. Intermittent advertisement screens appear with no content on it because the content is being blocked, this can cause confusion since it appears that the app is non-responsive.

 Love Balls	Passed	Game itself is not affected. Intermittent video advertisements are blocked and therefore no longer being shown, giving users uninterrupted flow between levels. However, users can no longer ask for hints since this feature requires the app to first show a video advertisement before showing the hint.
 Donuts Drift	Passed	Game itself is not affected. Advertisements which normally appear at the bottom of the screen are no longer being shown. Intermittent advertisement screens are also no longer being shown between games.
 Granny	Passed	Game itself is not affected. Intermittent video advertisements are blocked and therefore no longer being shown, giving users uninterrupted plays.







An interesting observation came out of this test – all advertisement contents, be it video or graphic, were delivered via HTTP and hence blocked. The original purpose of this study’s filtering proposal was to address security issue by blocking insecure HTTP traffic, this test has shown that it has inadvertently introduced a beneficial side effect – advertisements are now blocked, giving gamers uninterrupted plays for most of the games. Flip the Gun was the only game which was still popping up the advertisement screen, but with empty content.



This finding is in line with what Kaspersky researchers have discovered recently (Osborne, 2018), where a number of popular apps were found to be using insecure advertising Software Development Kits (SDKs). These SDKs, which use HTTP protocol for delivering contents as well as collecting user data, are often offered free to developers.

Reading News and Checking Weather

Next, some of the top Australian news (Nielsen, 2018) and weather websites were put to the test. Chrome browser app was once again used for conducting this test. Results are documented in Table 9 below.

Table 9: Results of Visiting Top Australian News and Weather Websites Using HTTPS

Website	URL	Result	Notes
 News.com.au	https://news.com.au https://www.news.com.au	Failed Failed	https://news.com.au raises an SSL error with no certificate matching hostname ‘news.com.au’, while https://www.news.com.au redirects to http://www.news.com.au
 Nine News	https://nine.com.au https://www.nine.com.au	Passed Passed	https://nine.com.au redirects to https://www.nine.com.au
 ABC News	https://abc.net.au/news https://www.abc.net.au/news	Failed Failed	https://abc.net.au/news is refusing connection altogether, while https://www.abc.net.au/news raises an SSL error with no certificate matching hostname ‘www.abc.net.au’
 Daily Mail Australia	https://dailymail.co.uk/auhome https://www.dailymail.co.uk/auhome	Failed Failed	https://dailymail.co.uk/auhome raises an SSL error with no certificate matching hostname ‘dailymail.co.uk’, while https://www.dailymail.co.uk/auhome also raises an SSL error with no certificate matching hostname ‘www.dailymail.co.uk’
 Yahoo7 News	https://au.news.yahoo.com	Passed	This works without any issues.
 Bureau of Meteorology	https://bom.gov.au https://www.bom.gov.au	Failed Failed	https://bom.gov.au is refusing connection altogether, while https://www.bom.gov.au shows a

			blank page titled 'Service Unavailable'
 Weatherzone	https://weatherzone.com.au https://www.weatherzone.com.au	Failed Failed	https://weatherzone.com.au raises an SSL error with no certificate matching hostname 'weatherzone.com.au', while https://www.weatherzone.com.au redirects to http://www.weatherzone.com.au
 AccuWeather	https://www.accuweather.com/en/au/australia-weather	Passed	This works without any issues.

Interestingly, this test has exposed a number of popular news and weather websites that do not support HTTPS. First of all, a number of these domain names do not have SSL certificate set up (news.com.au, www.abc.net.au, dailymail.co.uk, www.dailymail.co.uk, and weatherzone.com.au). Secondly, some are redirecting to their HTTP counterpart (https://www.news.com.au and https://www.weatherzone.com.au). The rest are refusing connection (https://abc.net.au/news and https://bom.gov.au) or denying the service (https://www.bom.gov.au).



One explanation for why a number of these websites are reluctant to support HTTPS is that their resources (such as graphic, JavaScript and CSS files) are delivered by some third-party CDN (Content Delivery Network) over HTTP for performance reasons. If a webpage is delivered over HTTPS but its resources are delivered over HTTP – also known as mixed content – it will get blocked by web browsers (van Bergen, 2018). Additionally, many of these websites are also heavily advertisement sponsored. Most advertisement networks, as highlighted in the gaming test, are delivering their contents over HTTP, also for performance reasons. For these reasons, operators of these websites have little to no incentives to move on to HTTPS.

Regarding the two government-funded websites, ABC News and Bureau of Meteorology, where the advertisement is not a deciding factor, it is disappointing to learn of their lack of HTTPS support. One can only assume that it is due to lack of funding for upgrades that are not justifiable.

Navigation

In this test, navigation apps were put to the test. Results are documented in Table 10.

Table 10: Results of Using Navigation Apps




App	Result	Notes
 Google Maps	Passed	App works flawlessly.
 Waze	Passed	App works flawlessly.

No issues were encountered while using these apps. Locations are obtained from GPS, so this aspect will not be impacted by the filtering solution. The apps do require to communicate with their backend servers and no issues were found, this confirms that they are communicating over HTTPS.

Banking

Some banking apps were put to the test next. Results are tabulated in Table 11 below.

Table 11: Results of Using Banking Apps



App	Result	Notes
 Westpac	Passed	App works flawlessly.
 Bankwest	Passed	App works flawlessly.
 ING	Passed	App works flawlessly.

The test has confirmed that all these banking apps are communicating with their backend over HTTPS, as they all work flawlessly in every aspect. This is something that is expected of financial institutions when dealing with sensitive data.

Taking Pictures

This test investigates whether user experience of taking pictures is affected in any way. Two apps were used in this test, their results are documented in Table 12 below.

Table 12: Results of Using Navigation Apps





App	Result	Notes
 Camera app	Passed	Works flawlessly.
 Google Photos	Passed	Works flawlessly. Photos were synchronised to and from the cloud seamlessly. Notifications work as normal.

As expected, the test shows that the Camera app is not affected in any way, since it works independently of network connectivity. Google Photos has also shown to work well, photos taken on the phone were uploaded to the cloud as usual.

Using Other Common Apps

In this test, some other apps are being investigated, the results are tabulated in Table 13 below.

Table 13: Results of Using Common Apps

App	Result	Notes
 Google Play Store	Passed	Works flawlessly, including downloading and installing apps.
 Ebay	Passed	Works flawlessly. Notifications work as normal.
 Gumtree	Passed	Works flawlessly. Notifications work as normal.
 ABC	Failed	On the main screens, only textual contents are showing, all graphics contents are not displaying. Furthermore, when an article is clicked, the page displays an error saying “Webpage not available” due to the app trying to load an HTTP content.

CONCLUSION

Mobile devices are constantly making connections with remote services, many of which are insecure with cleartext contents visible to third-parties while in transit. As a result, sensitive information is being exposed, and the device is made vulnerable to man-in-the-middle attacks. If all insecure outbound traffic can be blocked on the device, then such attacks can be mitigated.

This research has proved that a non-rooted and on-device filtering solution is feasible on an Android device, without the need of an external VPN service. It was achieved by using the VpnService class provided by the Android SDK. Outbound packets are routed to the filtering app through the virtual network interface, giving the app the ability to block HTTP packets only.

Such filtering solution, as shown in the speed tests, is very performant with little impact on the network speed, while day-to-day mobile usage tests have revealed a number of interesting things. First of all, there is no noticeable drain on power consumption, even though Android might incorrectly attribute power and network usage of other apps to NetGuard (“NetGuard Frequently Asked Questions (FAQ)”, 2018).

Secondly, app usage is mostly unaffected. Apps that rely on HTTP were quickly exposed, with contents missing and failing to load, such as ABC iView, SBS on Demand, and ABC apps. All gaming apps tested were affected but only in the delivery of advertisement contents, not in the game plays.

The area which is impacted the most is the browsing experience. This is due to a number of websites still failing to support HTTPS, despite Google’s plan of shaming and downgrading ranking on HTTP-only websites.

FUTURE WORK

As discussed earlier, the implementation of the filtering solution taken in this research blocks TCP packets destined for remote host’s port 80 only, HTTP connections to other ports will not be blocked. Additionally, all other network

protocols, such as UDP, as well as all other cleartext TCP protocols not destined for port 80, will also not be blocked. One recommendation for further research is to make required modifications in NetGuard such that it examines content in the transport or application layer, rather than just in the network layer, to determine whether the traffic should be blocked.

Another area for further research is to extend this work to iOS. iOS NetworkExtension SDK provides a class called NEPacketTunnelProvider (“NEPacketTunnelProvider – Create a principal class for a Packet Tunnel Provider app extension”, n.d.) which can be used for this purpose.

REFERENCES

- Android SDK VpnService. (n.d.). Retrieved from <https://developer.android.com/reference/android/net/VpnService.html>
- Android Studio. (n.d.). Retrieved from <https://developer.android.com/studio/>
- Bahajji, Z. A. & Illyes, G. (2014, Aug 6). HTTPS as a ranking signal. Retrieved from <https://webmasters.googleblog.com/2014/08/https-as-ranking-signal.html>
- Barnes, R. (2015, Apr 30). Deprecating Non-Secure HTTP. Retrieved from <https://blog.mozilla.org/security/2015/04/30/deprecating-non-secure-http/>
- Bokhorst, M. (2018). NetGuard A simple way to block access to the internet per application. Retrieved from <https://www.netguard.me>
- Chong, K. (2018, May 14). NetGuard Modified to Block HTTP Traffic. GitHub repository <https://github.com/chongkev/NetGuard>
- Claburn, T. (2018, Feb 8). From July, Chrome will name and shame insecure HTTP websites. Retrieved from https://www.theregister.co.uk/2018/02/08/google_chrome_http_shame/
- Conger, K. (2016, Jun 15). Apple will require HTTPS connections for iOS apps by the end of 2016. Retrieved from <https://techcrunch.com/2016/06/14/apple-will-require-https-connections-for-ios-apps-by-the-end-of-2016/>
- Electronic Frontier Foundation. (n.d.). *Encrypting the Web*. Retrieved from <https://www.eff.org/encrypt-the-web>
- Encrypt All the Things. (n.d.). Retrieved from <https://encryptallthethings.net>
- Google Developers. (2014, Jun 26). *Google I/O 2014 - HTTPS Everywhere*. [Video file]. Retrieved from <https://www.youtube.com/watch?v=cBhZ6S0PFCY>
- Ikram, M. & Kaafar, M. A. (2017, Oct. 30 – 2017, Nov. 1). A first look at mobile Ad-Blocking apps. Paper presented at the 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA).
- Jingjing, R., Ashwin, R., Martina, L., Arnaud, L., David, C. (2016) ReCon: Revealing and Controlling Privacy Leaks in Mobile Network Traffic. Proceedings of the 14th Annual International Conference / Mobile Systems. ACM, 2 Penn Plaza, Suite 701, New York, NY 10121-0701, USA.
- NEPacketTunnelProvider – Create a principal class for a Packet Tunnel Provider app extension. (n.d.). Retrieved from <https://developer.apple.com/documentation/networkextension/nepackettunnelprovider>
- NetGuard Frequently Asked Questions (FAQ). (2018, Apr 14). Retrieved from <https://github.com/M66B/NetGuard/blob/master/FAQ.md>
- Nielsen. (2018) Discover What Australians are Consuming Online Daily for Current Events and Global News Content. Retrieved May 5 from <http://www.nielsen.com/au/en/top10s.html>
- Osborne, C. (2018, Apr 17). Mobile apps transmit unencrypted user data due to insecure SDKs. Retrieved from <https://www.zdnet.com/article/mobile-apps-transmit-unencrypted-user-data-due-to-insecure-sdks/>
- Ozsoy, M. (2016). Frequency based advertisement blocking on Android mobile devices using a local VPN server. California State University, Long Beach. Retrieved from <http://search.proquest.com/docview/1851236425?accountid=10351>

- REVEALED: Top uses of our smartphones - and calling doesn't even make the list. (2017, Mar 13). Retrieved from <https://www.express.co.uk/life-style/science-technology/778572/Smartphone-phone-common-reason-use-call>
- Rutherford, S. (2017, Dec 19). Firefox May Soon Start Publicly Shaming Sites with Crappy Security. Retrieved from <https://www.gizmodo.com.au/2017/12/firefox-may-soon-start-publicly-shaming-sites-with-crappy-security/>
- Schechter, E. (2018, Feb 8). A secure web is here to stay. Retrieved from <https://security.googleblog.com/2018/02/a-secure-web-is-here-to-stay.html>
- Schechter, E. (2016, Sep 8). Moving towards a more secure web. Retrieved from <https://security.googleblog.com/2016/09/moving-towards-more-secure-web.html>
- Shuba, A. (2016). AntMonitor: A System for Mobile Network Monitoring. University of California. Retrieved from <http://escholarship.org/uc/item/0s02972x>
- Siciliano, R. (2017, Aug 18). How Does a VPN Protect My Computer and Privacy? Retrieved from <https://www.thebalance.com/how-vpn-protects-your-computer-and-privacy-4148267>
- Song, Y. & Hengartner, U. (2015). PrivacyGuard: A VPN-based Platform to Detect Information Leakage on Android Devices. Paper presented at the Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices, Denver, Colorado, USA.
- Speedtest by Ookla. (n.d.). Retrieved from <https://play.google.com/store/apps/details?id=org.zwanoo.android.speedtest>
- The most common usage of a mobile phone. (2015, Dec 7). Retrieved from <https://www.samsung.com/ae/discover/your-feed/the-most-common-usage-of-a-mobile-phone/>
- Thierer, T. (2017, Apr 11). Android O to drop insecure TLS version fallback in HttpURLConnection. Retrieved from <https://android-developers.googleblog.com/2017/04/android-o-to-drop-insecure-tls-version.html>
- Top Sites in Australia. (2018). Retrieved May 5 from <https://www.alexa.com/topsites/countries/AU>
- Trivedi, N. & Das, M. L. (2017). MalDetec: A Non-root Approach for Dynamic Malware Detection in Android. Information Systems Security : 13th International Conference, ICISS 2017, Mumbai, India, December 16-20, 2017, Proceedings (pp. 231-240): Cham : Springer International Publishing : Springer.
- van Bergen, J. (2018, Apr 9). What Is Mixed Content? Retrieved from <https://developers.google.com/web/fundamentals/security/prevent-mixed-content/what-is-mixed-content>
- VPN blocking. (2018, Apr 1). In *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/VPN_blocking
- Vyas, T. & Dolanjski, P. (2017, Jan 20). Communicating the Dangers of Non-Secure HTTP. Retrieved from <https://blog.mozilla.org/security/2017/01/20/communicating-the-dangers-of-non-secure-http/>
- Wagenseil, P. (2014, Jul 18). HTTP Must Die, Security Experts Tell Hackers. Retrieved from <https://www.tomsguide.com/us/http-must-die,news-19188.html>