2018

Bringing defensive artificial intelligence capabilities to mobile devices

Kevin Chong
Edith Cowan University

Ahmed Ibrahim
Edith Cowan University

DOI: 10.25958/5c526a5866687
Chong, K., & Ibrahim, A. (2018). Bringing defensive artificial intelligence capabilities to mobile devices. In *proceedings of the 16th Australian Information Security Management Conference* (pp. 41-50). Perth, Australia: Edith Cowan University.

BRINGING DEFENSIVE ARTIFICAL INTELLIGENCE CAPABILITIES TO MOBILE DEVICES

Kevin Chong, Ahmed Ibrahim
School of Science, Edith Cowan University, Perth, Australia
kkchong0@our.ecu.edu.au, ahmed.ibrahim@ecu.edu.au

Abstract

Traditional firewalls are losing their effectiveness against new and evolving threats today. Artificial intelligence (AI) driven firewalls are gaining popularity due to their ability to defend against threats that are not fully known. However, a firewall can only protect devices in the same network it is deployed in, leaving mobile devices unprotected once they leave the network. To comprehensively protect a mobile device, capabilities of an AI-driven firewall can enhance the defensive capabilities of the device. This paper proposes porting AI technologies to mobile devices for defence against today's ever-evolving threats. A defensive AI technique providing firewall-like capability is being presented. The possibility of tracing both outbound and inbound network packets to a specific mobile app is being explored. This ability of isolating network traffic to specific apps plays an important part in data cleansing for use in AI. With such isolated network data, accurate models can be trained up individually for each app. It would then be possible to build up a baseline of what normal traffic looks like for an app and any deviation from this baseline can be detected. In our proposed model, anomalies deemed malicious can be blocked for a specific app while leaving others unaffected.

Keywords
Artificial intelligence, machine learning, mobile device, on-device, firewall, intrusion detection, security

INTRODUCTION

Cyber criminals are actively on the lookout to compromise all kinds of network enabled devices and mobile devices are no exception. Even though official app stores have their own procedures of scanning apps before publication, malicious apps still manage to sneak through (Newman, 2017; Dassanayake, 2018; Tee & Zhang, 2018; Ramel, 2018; Leyden, 2017; XcodeGhost, 2018). The damages these apps can potentially cause can be mitigated if malicious communication initiated or received can be identified, isolated, and cut-off.

As effective as they are in filtering unwanted network traffic, firewalls can only protect devices inside the network they are deployed for. They cannot protect mobile devices once they leave the perimeter. Furthermore, attacks are increasingly getting more dynamic in nature, many of which are now being further sophisticated with the usage of Artificial Intelligence (AI) (Dutt, 2018; Elazari, 2017; Patterson, 2018). Traditional firewalls, which are typically configured by static rules are proving to be ineffective against such threats.

This paper looks at how these threats can be mitigated on a mobile device. The proposed solution explores how defensive filtering AI capabilities can be brought onto mobile devices without the reliance on external firewalls. Such an approach can protect the mobile device at all times, regardless of which network it is connected to.

AI-DRIVEN FIREWALLS

AI-driven firewalls have been gaining popularity recently (Dilek, Çakır & Aydın, 2015). Unlike traditional firewalls where the header of a network packet is examined against a set of statically configured rules, this new breed of firewalls uses Machine Learning (ML) algorithms to examine packets holistically. They are trained from real-world data, allowing them to identify abnormal activities in a variety of situations (Jyothsna & Prasad, 2011). Furthermore, the ability to self-learn in some systems make them effective in detecting even unknown threats (Schneier, 2018).

There are a number of different techniques used by various AI-driven firewall vendors, many of them are anomaly-based in their approaches (Buczak & Guven, 2016). These systems are trained using ML algorithm to establish what a normal activity profile and any exceptions to this will be detected. This anomaly-based approach typically

has low false negative rate, making it particularly effective against zero-day attacks (Anwer, Farouk & Abdel-Hamid, 2018).

Each ML algorithm employs its own predictive model for their classification. The type of predictive model used affects feature selection and dimensionality reduction. Some of the predictive models used by AI-driven firewalls are Decision Tree and Naïve Bayes (Bhamare, Salman, Samaka, Erbad, & Jain, 2016).

As effective as they are in defending the kind of threats introduced by malicious apps, many of these standalone AI-driven firewalls were designed for high-end hardware, with little consideration given to platforms of limited resource. For this reason, these firewalls are not viable on mobile hardware.

AI TECHNOLOGIES ON MOBILE DEVICES

As most AI technologies have been designed to run on high-end hardware, they are unsuitable for deployment on mobile devices in their unmodified form. In recent years, however, we have started to see a trend for AI companies, such as Google and Apple, investing more of their resources in finetuning their algorithms for mobile devices. This is not surprising since mobile devices have now become an indispensable part of day-to-day life of most people and that there are various applications for AI, such as face recognition, personal assistant and augmented reality just to name a few. Apple has designed two of their latest Bionic chips, A11 and A12, to power their AI technology ("Apple A11", 2018; "Apple A12", 2018). Across the industry, more of the Graphical Processing Unit (GPU) usage is now being allocated for AI computation. This is the beginning of a shift we are seeing in making AI central to every interaction we have with mobile devices.

The push is not just in getting the benefits of AI to mobile users, but also in getting the entire algorithm executing on the device. The advantages of this on-device self-sufficiency approach are ("Neural Networks API", 2018):

1. Latency reduction – data is not sent over the network for results, this is critical for time sensitive applications such as real-time speech processing.
2. Availability while off-line – AI features are always available even when the network is unavailable.
3. Privacy is preserved – user's data is kept private since nothing is sent over the network.
4. Server cost reduction – no servers are needed since computations are performed on device.

The following section discusses some of the AI technologies that are currently available on mobile devices.

Android Neural Networks API

The Android mobile Operating System (OS) offers the Android Neural Networks API (NNAPI) as part of their native Software Development Kit (SDK). The NNAPI is a new offering from Google supporting devices running Android 8.1 and newer. It provides a base layer of functionality for higher-level AI frameworks, such as TensorFlow, to build and train decision models ("Neural Networks API", 2018).

NNAPI's main purpose is to efficiently distribute the computation workload across various on-device processors, such as GPUs, Digital Signal Processors (DSPs), general Central Processing Units (CPUs) and any other specialised neural network processing chips. Figure 1 shows the system architecture of NNAPI.

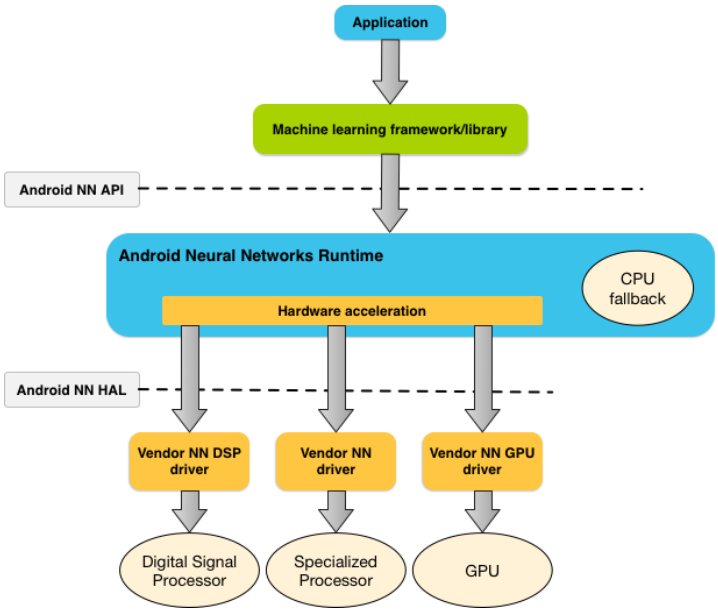

Figure 1: Android NNAPI Architecture ("Neural Networks API", 2018)

Core ML

Core ML is an ML framework from Apple targeting iOS. It supports extensive deep learning with over thirty layers such as tree ensembles, Support Vector Machines (SVMs) and generalised linear models. Core ML maximises computation across CPU and GPU on mobile devices to provide best possible performance and power efficiency ("Machine Learning – Get Ready for Core ML 2", 2018).

TensorFlow Lite

TensorFlow Lite is a lightweight version of TensorFlow, an open source ML framework for high performance computation by Google. TensorFlow Lite has been fine tuned for Android and iOS devices. On Android, it leverages NNAPI to utilise all of the device hardware acceleration.

The developers of TensorFlow Lite had redefined its own file format, optimised its interpreter and revamped how memory gets allocated. As a result, it is possible to perform inference on the device without consuming excessive resources, while keeping applications lean and fast. Figure 2 shows the architectural design of TensorFlow Lite ("Introduction to TensorFlow Lite", 2018).

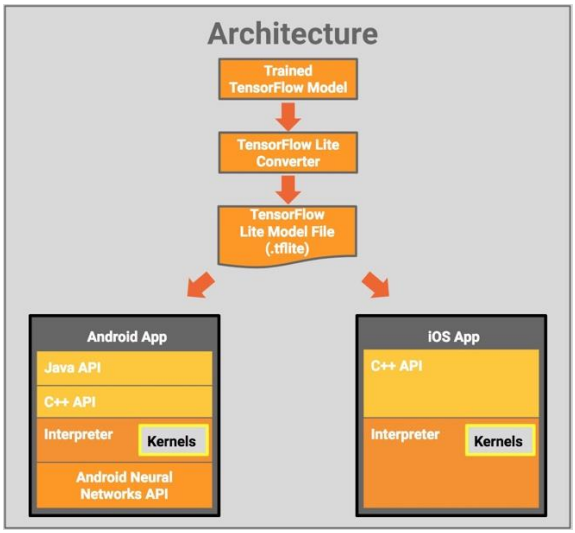

Figure 2: Architectural Design of TensorFlow Lite ("Introduction to TensorFlow Lite", 2018)

Lack of Focus in Malicious Network Traffic Detection

While the aforementioned AI technologies are suitable for mobile hardware, they lack focus in network filtering. Much of the interests from these vendors are on improving the mobile user experience, such as personal assistance, image recognition and sensory data processing, rather than on detecting malicious network traffic.

FIREWALL ON MOBILE DEVICES

Firewalls are low-level applications, they operate mostly at the Network layer (Layer 3) of the Open Systems Interconnection (OSI) model. This presents a challenge to mobile devices since both Android and iOS OSes do not provide app developers access to the low-level raw socket interface – raw sockets are only available for "rooted" or "jailbroken" devices (rooting or jailbreaking refers to a procedure whereby restrictions imposed by the OS vendor are removed and therefore effectively acquiring complete administrative privileges). The other challenge is that firewalls typically examine every outbound and inbound network packet, and without dedicated hardware for such purposes, this can easily exhaust the battery on a mobile device.

The solution to the latter challenge is to optimise the code, ensuring that it is not doing any unnecessary and costly computation. To reduce overhead, developers should look into implementing these executions in low level language such as C/C++.

Firewall Implementation on Android

To tackle the first challenge, Android SDK provides apps a virtual network interface where all traffic gets routed through without needing root access. This interface, provided via API class VpnService ("Android SDK VpnService", n.d.), is essentially a file descriptor, where each "read" from the descriptor retrieves an outbound Internet Protocol (IP) packet and each "write" injects an inbound IP packet. A number of past studies requiring 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 firewall app examine every outgoing packet. If the firewall infers that a particular outgoing packet should be blocked, it will simply drop it, otherwise, it will make necessary connections with the packet's target and route it to its rightful destination. On the reverse side, when the firewall app gets incoming traffic from these opened connections, it will route it back into the virtual network interface.

Firewall Implementation on iOS

Similar to Android's VpnService class, iOS provides a class called NEPacketTunnelProvider ("NEPacketTunnelProvider – Create a principal class for a Packet Tunnel Provider app extension", n.d.). In addition, iOS's NetworkExtension SDK offers the ability to route only DNS traffic to the virtual interface instead of every network packet. This gives the ability for the firewall to block traffic by domain names.

How Network Traffic is Blocked

As illustrated in Figure 3, this is how network traffic initiated from an app is blocked by the firewall (Chong, Malik & Hannay, 2018):

1. An app initiates a network connection with a remote host at example.com.
2. Android/iOS routes the packet to the virtual network interface provided by VpnService/ NEPacketTunnelProvider respectively.
3. Firewall app receives the packet from the virtual network interface.
4. Firewall app examines the packet.
5. Firewall app infers that this packet should be dropped, it does nothing further with it, essentially blocking it from leaving the device.

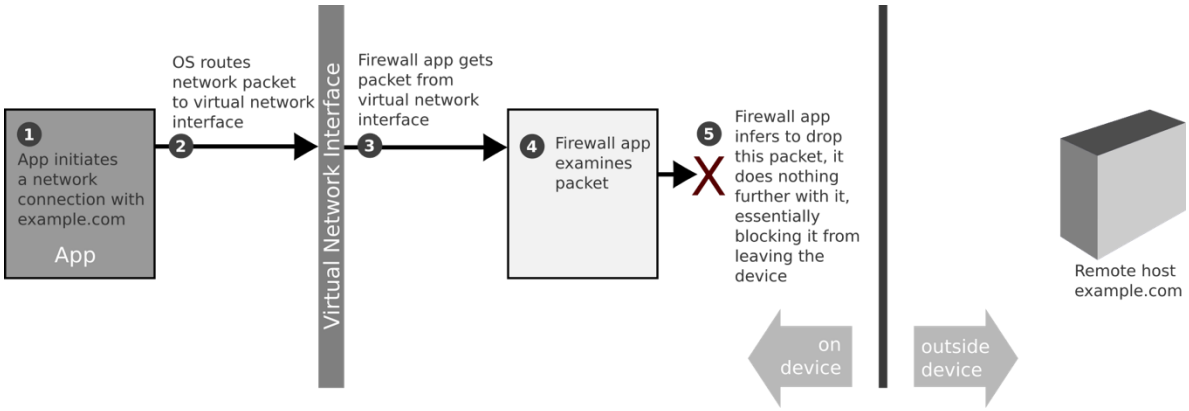

Figure 3: How Network Traffic is Blocked by the Firewall App, adopted from Chong, Malik and Hannay (2018)

How Network Traffic is Allowed

The following Figure 4 illustrates how network traffic is allowed (Chong, Malik & Hannay, 2018):

1. An app initiates a network connection with a remote host at example.com.
2. Android/iOS routes the packet to the virtual network interface provided by VpnService/ NEPacketTunnelProvider respectively.
3. Firewall app receives the packet from the virtual network interface.
4. Firewall app examines the packet.
5. Firewall app infers that this packet should be not be dropped, it establishes a connection with the remote host.
6. Once the connection is established, firewall app routes the traffic to the remote host.
7. Firewall app receives incoming traffic from the remote host via the connection.
8. Firewall app routes incoming packets into the virtual network interface.
9. Android/iOS routes incoming packets to the app that initiated the network connection.

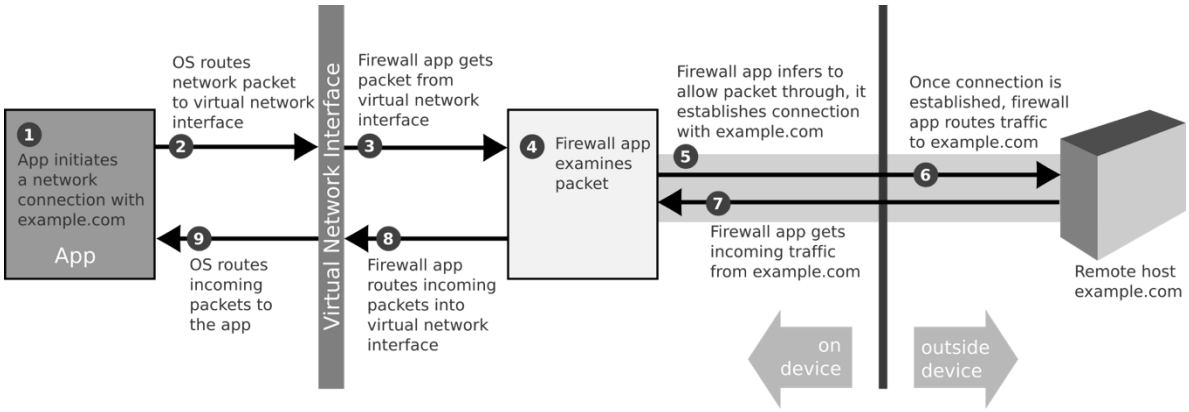

Figure 4: How Network Traffic is Allowed by the Firewall App, adopted from Chong, Malik and Hanna (2018)

Lack of Focus in Deploying AI Algorithms

While there are a number of on-device firewall apps using the technique discussed above ("NoRoot Firewall FAQ", n.d.; "NetGuard – A simple way to block access to the internet per application", 2018; "Mobiwol No Root Firewall", 2014), none of these are using AI algorithms in deciding how to block packets. The next section discusses the integration of mobile AI technologies into these firewall apps.

AI-DRIVEN FIREWALL ON MOBILE DEVICES

The approach of the AI-driven firewall proposed in this paper involves the following stages:

1. Collection of training and testing data.
2. Training and testing the AI framework.
3. Optimisation of trained model for deployment.
4. AI integration with firewall app.

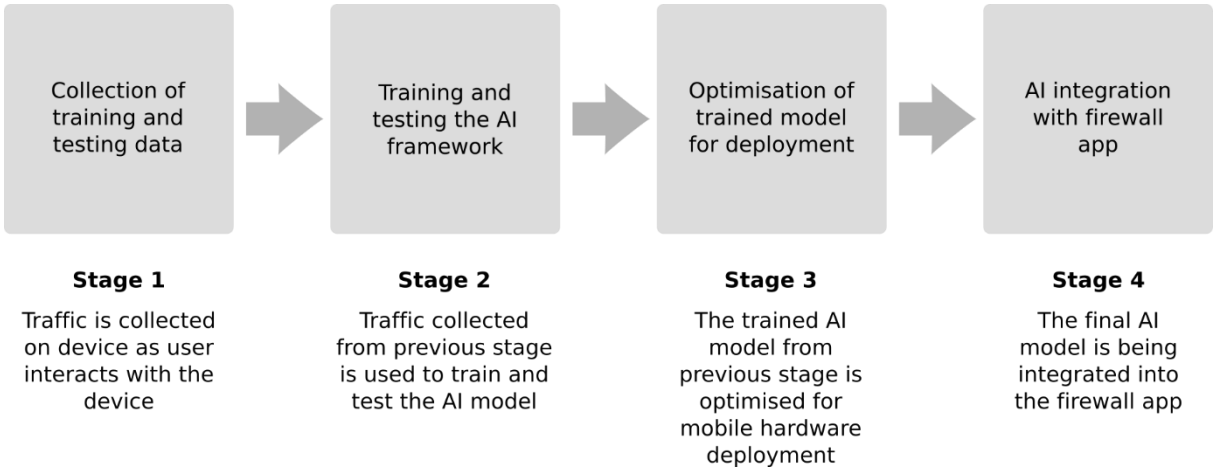

Figure 5: The four stages of AI-driven firewall

Collection of Training and Testing Data

The data ultimately used by the AI-driven firewall for inference will be network traffic, hence realistic network traffic from a mobile device needs to be collected for training purposes. The first type of traffic that needs to be collected is normal traffic. This type of data will be used to train the ML framework in establishing a baseline pattern for typical traffic. Once the model has learned what normal traffic looks like, it will then be able to detect abnormal traffic with some acceptable degree of certainty.

Normal traffic is relatively easy to collect. As explained earlier, all traffic from the device gets routed to the virtual network interface. During the capture mode, the firewall app having access to the interface, simply captures all packets without any examination. All that is required is to use a mobile device in a typical fashion while the traffic is being captured.

The second type of traffic that needs to be collected is traffic that mimics a malicious activity. This type of traffic needs to be manufactured. This would involve purposefully running apps which are known to be malicious and capturing their traffic.

It is important to note that during the capture session, the capturing firewall app itself should restrain from using the network, otherwise it will interfere with regular traffic it is trying to capture. Captures should be written straight to device storage with no processing, preferably in pcap format (“pcap”, 2018) as this is the standard for network captures.

Training and Testing the AI framework

Training and testing are a computationally intensive process, which should not be performed on the device. Captured network data will need to be offloaded from the device at the completion of a capture session. Before training and testing is carried out, the data needs to be split into training set and testing set.

Part of the work involved in the training and testing procedure is to find the most suitable AI model for the intended purpose. This process might even involve trial-and-error until a suitable model is trained.

If the model yields an acceptable result from the testing set, then the model is considered to be acceptable. Otherwise, further adjustments will have to be made for retraining and retesting.

Optimisation of Trained Model for Deployment

To be suitable for execution on mobile hardware, AI models often need to be optimised. In addition, depending on which AI framework is chosen for the task, it may be required to convert the trained model to one that is suitable to be consumed by the inference framework on the device.

AI Integration with Firewall App

The final stage involves integrating the AI framework into the firewall app. Outgoing packets that get routed through the virtual network interface will be fed to the trained model. By using the trained model, a decision is made as to whether a packet should be blocked or allowed.

If the packet is to be blocked, the firewall app does nothing further with the packet. Otherwise, the firewall app establishes a connection with the packet's destination host and forwards the packet to it. The firewall app also needs to listen for incoming traffic from the destination host and redirects the packet back into the virtual network interface. The rest of the flow is taken care of by the OS as the packet is routed to the appropriate app.

Since the firewall is not driven by a set of static rules, but by generalisation by an ML trained model, it is expected to handle threats which it had not seen before. Rather than examining the header of individual packets and their contents, the AI-driven firewall analyses the sequence of packets in their context to detect any alarming deviation from normal traffic pattern. It is this ability that would make it a better defensive firewall against unprecedented attacks.

Isolating Network Traffic to Individual Apps

Android's Linux-based kernel uses a special filesystem called the proc filesystem (procfs) to present information regarding running processes. The files in the directory `/proc/net` contain information regarding the network stack ("procfs", 2018). For example, the files `/proc/net/tcp` and `/proc/net/udp` contain information about existing TCP and UDP connections respectively, where the UID (User ID) can be retrieved for an existing connection with known destination IP address and port, and source IP address and port.

Every running app on Android is sandboxed and is given a unique UID. The PackageManager class in the Android SDK provides the ability to retrieve the app name and the app package identifier (also known as app ID) given a UID, via getNameForUid and getPackagesForUid methods ("Android SDK PackageManager", 2018).

Given the above, the firewall app can track network traffic to individual apps. As the virtual network interface feeds IP packets to the firewall app, the firewall app will then be able to retrieve IP addresses' and ports' information from these packets, and with this information, the firewall app can infer which app is making these network connections. NetGuard uses this very technique to block internet access for specified apps ("NetGuard – A simple way to block access to the internet per application", 2018).

This ability to isolate network traffic to individual apps has given the AI framework big advantages. Firstly, it has provided a way of data cleansing. The AI model is no longer being presented with huge amount of traffic which might seem unrelated to each other. Instead, each network packet can now be traced to an individual app. With this new dimension, the framework can now analyse the jungle of network traffic in isolation, for example, it knows that a particular packet comes from a particular app, so it will only analyse it in one particular context.

Secondly, it has provided a way for the AI framework to deploy individually trained models instead of one large monolithic model. Deploying multiple models has the advantage of tuning and training specific models fit for specific apps. For example, it is possible to train a particular model targeting a specific app, such that it is optimised in detecting anomaly traffic pattern made from that particular app. Since these models are now streamlined to handle much less traffic in their given context, their complexity and size can be reduced dramatically. This is a welcome optimisation given the limited resources that are available on mobile hardware. Furthermore, because of the reduction in complexity, the time needed to train the AI model will also be reduced.

Thirdly, the prediction made by the AI framework will likely be improved. This isolation technique treats network traffic in a sandboxed manner, whereby traffic is divided into a context for each app. The inference model of an individual app deals with much less traffic, but in a context that is much more relevant for the app, it is this scenario of an improved relevance of the input data that will give rise to the accuracy of the prediction.

However, it should be noted that Android Pie, which was officially released in August 2018 ("Android Pie", 2018), has restricted regular apps access to the proc filesystem (Chirgwin, 2018; Rahman, 2018), the technique being

discussed in this section might no longer be possible from this Android version onward. On iOS, there is no known way for a regular app to access this type of network information.

CURRENT AND FUTURE CHALLENGES

Limitation in mobile hardware to crunch through complex AI algorithm in real time remains one of the biggest challenges today. It is not just the amount of network data that needs to be fed through to the algorithm, but also its speed. Such a computationally intensive algorithm requires a fast CPU and good amount of constant supply of power, both of which have been an issue for mobile hardware to this day. Recent research conducted by a Swiss university has revealed that there are areas still lacking, where frameworks are incomplete and chipsets not supporting various neural networks with unreliable results (Ignatov et al., 2018; Ray, 2018).

Another challenge is finding the best combination of mobile AI technology and model for the anomaly-based approach proposed in this paper. Furthermore, unlike other AI-driven firewalls, the proposed technique deals with traffic that is isolated to individual apps, additional research needs to be carried out to find the most suitable algorithm for such an approach, or perhaps to categorise apps that inhibit common attributes and behaviour.

As presented in this paper, models are trained from individual apps from gathering their network activity. With high frequency of app updates these days, this can present a challenge where training needs to be kept up with each update. It could potentially become impractical to keep up with the pace of these updates.

A central aspect for training a reliable ML model is to have good data to define the baseline. Thus, anomalies detected would be only as good as the baseline defined with help of benign network traffic generated during training. This can be addressed during experimentation, but poses several unknown variables when it comes to real-world usage of the mobile device.

The attacks that have surfaced in recent years are gaining in sophistication. It is known that attackers are now turning to AI in their offence. In this regard, the challenge for today and the future is to keep pace such that the defensive strategy is always ahead of any attack. Defensive AI is looking promising as a strategy to combat against offensive AI, but is it the only feasible option? There are discussions about employing offensive AI for defence purposes (Uren, Hogeveen & Hanson, 2018; Hanson & Uren, 2018). Is this a possible strategy and is this even ethical?

ETHICAL ISSUES

The core technology being proposed in this paper uses an on-device firewall app which intercepts and analyses every outbound and inbound packet. Since a mobile device has become so personal to its user, it is possible to build an accurate profile and even habits of the user from analysing the network traffic. The question that needs to be raised is, is this an ethical issue whereby user privacy is being invaded?

As discussed, network traffic needs to be collected to train the AI model. This training data, which is gathered while a user uses the device becomes the baseline of what is defined as normal traffic. The issue resulting from this is, what or who determines a user's usage habit more 'normal' than another user's? Has this become an ethical issue where one training set generated by one user is deemed more suitable for the job than another generated by another user?

The technology being presented in this paper has highlighted some surrounding ethical issues which possibly cannot be addressed by any legal system. Should such a technology be used at all before these grey areas are adequately addressed? This in itself is another ethical issue.

These ethical issues posed above are questions that could be argued on both sides. However, more research is needed to investigate both the technical and social dimensions with the increasing popularity and application of AI in various domains.

CONCLUSION

Vulnerable and malicious mobile apps are being installed on millions of devices despite efforts put in place to weed them out by official app stores. The threats posed by such malicious apps can be mitigated by cutting off unsolicited communication with their control-and-command server.

This research looked at employing defensive AI techniques to derive a firewall which runs locally on the mobile device. Anomaly-based ML approach was being proposed as a strategy against unknown threats. Challenges around deploying AI algorithm on mobile device were discussed, as some approaches aimed at working around limitations of the hardware. The technique of isolating network traffic to individual apps was proposed as an effective way to improve prediction and as a way to reduce complexity in the inference model. The firewall explored in this paper is a regular app which does not require special privileges offering constant protection regardless of which network it is connected to. Our future work will look at implementing the proposed model and validating our approach through empirical evidence. We intend to publish our findings subsequently.

REFERENCES

Android Pie. (2018, Aug 31). In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Android_Pie

Android SDK PackageManager. (2018, Jul 2). Retrieved from https://developer.android.com/reference/android/content/pm/PackageManager

Android SDK VpnService. (2018, Jun 6). Retrieved from https://developer.android.com/reference/android/net/VpnService.html

Anwer, H. M., Farouk, M., & Abdel-Hamid, A. (2018, May 7). A framework for efficient network anomaly intrusion detection with features selection. 2018 9th International Conference on Information and Communication Systems (ICICS), Irbid, 2018, pp. 157-162. doi: 10.1109/IACS.2018.8355459

Apple A11. (2018, Sep 16). In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Apple_A11

Apple A12. (2018, Sep 16). In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Apple_A12

Bhamare, D., Salman, T., Samaka, M., Erbad, A., & Jain, R. (2016, Dec 19). Feasibility of Supervised Machine Learning for Cloud Security. 2016 International Conference on Information Science and Security (ICISS), Pattaya, 2016, pp. 1-5. doi: 10.1109/ICISSEC.2016.7885853

Buczak, A. L. & Guven, E. (2016). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications Surveys & Tutorials (Volume: 18, Issue: 2, Secondquarter 2016). doi: 10.1109/COMST.2015.2494502

Chirgwin, R. (2018, May 8). Android P to improve users' network privacy. Retrieved from https://www.theregister.co.uk/2018/05/08/android_p_will_improve_users_network_privacy

Chong, K., Malik, M. I., & Hannay, P. (in press). Mitigating Man-in-the-Middle Attacks on Mobile Devices by Blocking Insecure HTTP Traffic Without Using VPN. The Proceedings of 16th Australian Information Security Management Conference, 4-5 December, 2018, Edith Cowan University, Perth, Australia.

Dassanayake, D. (2018, Aug 6). ANDROID WARNING: Over a HUNDRED apps on Google Play Store FILLED with malware. Retrieved from https://www.express.co.uk/life-style/science-technology/999716/Android-warning-malware-apps-Google-Play-Store-Windows-malicious-software

Dilek, S., Çakır, H., & Aydın, M. (2015, Feb 12). Applications of Artificial Intelligence Techniques to Combating Cyber Crimes: A Review. Journal of Artificial Intelligence & Applications (IJAIA), Vol. 6, No. 1, January 2015. doi: 10.5121/ijaia.2015.6102

Dutt, D. (2018, Jan 10). 2018: the year of the AI-powered cyberattack. Retrieved from https://www.csoonline.com/article/3246196/cyberwarfare/2018-the-year-of-the-ai-powered-cyberattack.html

Elazari, K. (2017, Dec 28). Hackers are on the brink of launching a wave of AI attacks. Retrieved from https://www.wired.co.uk/article/hackers-ai-cyberattack-offensive

Hanson, F., & Uren, T. (2018, Apr 10). Australia's Offensive Cyber Capability. Retrieved from https://www.aspi.org.au/report/australias-offensive-cyber-capability

Ignatov, A., Timofte, R., Szczepaniak, P., Chou, W., Wang, K., Wu, M., ... & Van Gool, L. (2018). AI Benchmark: Running Deep Neural Networks on Android Smartphones. arXiv preprint arXiv:1810.01109.

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).

Introduction to TensorFlow Lite. (2018, Aug 8). Retrieved from https://www.tensorflow.org/mobile/tflite

Jyothsna, V. & Prasad, V. V. R. (2011, Sep). A Review of Anomaly based Intrusion Detection Systems. International Journal of Computer Applications (0975 – 8887) Volume 28– No.7, September 2011.

Leyden, J. (2017, Jul 20). No one still thinks iOS is invulnerable to malware, right? Well, knock it off. Retrieved from https://www.theregister.co.uk/2017/07/20/ios_security_skycure

Machine Learning – Get Ready for Core ML 2. (2018). Retrieved from https://developer.apple.com/machine-learning

Mobiwol No Root Firewall. (2014). Retrieved from http://www.mobiwol.com

NEPacketTunnelProvider – Create a principal class for a Packet Tunnel Provider app extension. (n.d.). Retrieved from https://developer.apple.com/documentation/networkextension/nepackettunnelprovider

NetGuard – A simple way to block access to the internet per application. (2018). Retrieved from https://www.netguard.me

Neural Networks API. (2018, Aug 16). Retrieved from https://developer.android.com/ndk/guides/neuralnetworks

Newman, L.H. (2017, Sep 22). How Malware Keeps Sneaking Past Google Play's Defenses. Retrieved from https://www.wired.com/story/google-play-store-malware/

NoRoot Firewall FAQ. (n.d.). Retrieved from https://norootfirewall.weebly.com

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

Patterson, D. (2018, Aug 16). How weaponized AI creates a new breed of cyber-attacks. Retrieved from https://www.techrepublic.com/article/how-weaponized-ai-creates-a-new-breed-of-cyber-attacks

pcap. (2018, Jul 28). In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Pcap

procfs. (2018, Jun 16). In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Procfs

Rahman, M. (2018, May 6). Android P will finally restrict apps from monitoring your network activity. Retrieved from https://www.xda-developers.com/android-restrict-apps-monitor-network-activity

Ramel, D. (2018, Sep 13). Study: Open Source Software Contributes to Mobile App Vulnerabilities. Retrieved from https://adtmag.com/articles/2018/09/13/open-source-risks.aspx

Ray, T. (2018, Oct 5). AI on Android mobile phones still a work-in-progress. Retrieved from https://www.zdnet.com/article/ai-on-android-mobile-phones-still-a-work-in-progress

Schneier, B. (2018). Artificial Intelligence and the Attack/Defense Balance. IEEE security & privacy., 16(2), 96.

Shuba, A. (2016). AntMonitor: A System for Mobile Network Monitoring. University of California. Retrieved from http://escholarship.org/uc/item/0s02972x

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.

Tee, M.Y. & Zhang, M. (2018, May 9). Hidden App Malware Found on Google Play. Retrieved from https://www.symantec.com/blogs/threat-intelligence/hidden-app-malware-google-play

Uren, T., Hogeveen, B., & Hanson, F. (2018, Jul 4). Defining offensive cyber capabilities. Retrieved from https://www.aspi.org.au/report/defining-offensive-cyber-capabilities

XcodeGhost. (2018, Jun 21). In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/XcodeGhost