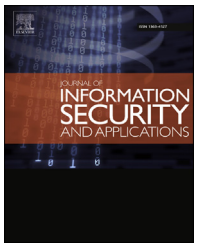
Available online at www.sciencedirect.com

SciVerse ScienceDirect

journal homepage: www.elsevier.com/locate/jisa



MASON: Mobile autonomic security for network access controls

*William M. Fitzgerald**, *Ultan Neville*, *Simon N. Foley*

Department of Computer Science, University College Cork, Ireland

Keywords:
Smartphone threats
Security configuration
Firewalls

ABSTRACT

Smartphones are on par with modern desktop environments in terms of operating system and hardware functionality. As a consequence, threats to desktop environments are also applicable to smartphones in addition to traditional threats to mobile phones. End-user management of security configurations that mitigate smartphone threats is complex and error-prone. As a consequence, misconfiguration of a security configuration may unnecessarily expose a smartphone to known threats. In this paper, a threat-based model for smartphone security configuration is presented. To evaluate the approach, a prototype Android security app, MASON, is developed to automatically manage firewall configurations on behalf of the end-user. A case study based on firewall access control demonstrates how automated firewall configuration recommendations can be made based on catalogues of countermeasures. These countermeasures are drawn from best-practice standards such as NIST 800-124, a guideline on cell phone and PDA security and NIST 800-41-rev1, a guideline on firewall security configuration.

1. Introduction

Modern smartphones with their processing power, operating systems and the wide variety of applications ("apps") are on a par with modern desktop environments (Shabtai et al., 2010). This has resulted in smartphones being used in a variety of domains from a personal device (such as for voice, Web browsing, Email and social media) to enterprise, medical and military domains (Wei et al., 2012). The technological advances and the usage of smartphones in a variety of domains are not without its security implications. In addition to traditional mobile phone threats, threats to desktop environments are also applicable to smartphones (Shabtai et al., 2010; Khadem, 2010; Chin et al., 2011). For example, Malware threats such as DroidDream (Balanza et al., 2011), a Android Market Trojan app used to maliciously root Android smartphones, are on the increase (Shabtai et al., 2010).

Smartphones may host a variety of security mechanisms such as anti-virus, app monitoring and firewalls. In practice, security mechanisms are either disabled or configured with an open access policy (Ruggiero and Foote, 2010). Configuration of smartphone security mechanisms, for example a firewall, is typically performed by non-technical end-users. As a consequence, an effective security configuration may be hampered by a poor understanding and/or management of smartphone application requirements. Misconfiguration, may result in the failure to adequately provide smartphone app services. For example, an overly-restrictive firewall configuration may prevent normal interaction of network-based apps. An overly-permissive firewall configuration, while permitting normal operation of the app, may leave the smartphone vulnerable to attack, for example, across open ports or malicious payloads.

Smartphones operate in mobile network environments and deploying a fixed security configuration for a global set of

* *Corresponding author.*
E-mail addresses: wfitzgerald@4c.ucc.ie (W.M. Fitzgerald), u.neville@4c.ucc.ie (U. Neville), s.foley@cs.ucc.ie (S.N. Foley).
2214-2126/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jisa.2013.08.001

threats is not practical. For example, a smartphone may in one scenario be connected to an enterprise WiFi network while in another be connected to an open access WiFi network or a 3g operator network. What may be considered a threat in one scenario may not be a threat in another. For example, a security configuration that permits a set of apps (such as gaming and social media apps) within a home network environment may not be permitted within an enterprise or teleworking environment. In a teleworking scenario, it is considered best practice to permit the use of "*a different brand of Web browser for telework*" and prohibit the use of the everyday Web browser (Scarfone and Souppaya, 2007). Thus, the deployment of smartphone security configurations must be dynamic in order mitigate the relevant threats within a given scenario.

This paper demonstrates the effectiveness of smartphone firewalls at mitigating or reducing the impact of known network-based threats. In terms of implementation, the focus is on the Android platform, a software framework for mobile devices such as smartphones (for example Nexus 4), tablet PC's (for example Nexus 10) and embedded devices (for example Neo-ITX). Note, while smartphone apps may provide their own end-to-end security, in accordance with for example (Evans et al., 2001), it is considered best practice to also restrict access at the smartphone firewall (Scarfone and Souppaya, 2007; Souppaya and Scarfone, 2012a; Jansen and Scarfone, 2008; Wack et al., 2002).

This paper is a revised and extended version of the paper in Fitzgerald et al., 2012). The contribution of this paper is as follows. A threat-based model that represents catalogues of best practice standards for smartphones is described. A case study for smartphone firewall configuration management is considered. This research extends the work in (Foley and Fitzgerald, 2011) where the firewall catalogues of best practice are smartphones centric and new catalogues of best practice for example NIST 800-114 (Scarfone and Souppaya, 2007) are developed. A prototype firewall app called MASON is developed for the Android platform (http://www.android.com/) to automatically manage firewall configurations on behalf of the non-expert end-user.

This paper is organised as follows. Section 2 outlines the smartphone threat landscape and the effectiveness of firewalls at mitigating threats. Section 3 provides an introduction to Linux iptables, the stock Android platform firewall. The challenges and complexity of smartphone firewall management are outlined in Section 4. A threat-based security model for smartphones is presented in Section 5. Section 6 outlines a set of best practice standards that are encoded within the security model. The implementation of the smartphone best practice catalogue is discussed in Section 7. MASON, an automated firewall app for the Android platform is discussed in Section 8. Related research is outlined in Section 9 and Section 10 concludes the paper.

2. Smartphone threat landscape

Unlike traditional mobile devices such as PDA's and feature phones, smartphones are on par with modern desktop environments in terms of operating system functionality, processing power, storage capacity, environmental sensors (for example GPS) and so forth. As a consequence, smartphone threats, for example, Malware, botnets, communication interception, DoS, port scanning, privacy leakage, resource exhaustion (for example battery) are on the increase (Shabtai et al., 2010; Khan et al., 2012; Wang et al., 2012; Felt et al., 2011; Landman).

Smartphone threat vectors include Bluetooth, WiFi, 3g, NFC and USB (Shabtai et al., 2010). For example, Bluebug (http://trifinite.org/) may be used to gain unauthorised access to phone contact lists and text messages on bluetooth-enabled smartphones. Smartphones connected to unsecured WiFi hotspots increase the threat of communication interception such MITM attacks and password eavesdropping (Landman).

Smartphones have a number of security mechanisms, for example application sandboxes, application permission management, data encryption, remote management, antivirus and firewalls, that are used to protect smartphones and their data from unauthorised access (Shabtai et al., 2010; Husted et al., 2011).

Section 2.1 demonstrates the effectiveness of firewalls at mitigating or reducing the impact of known network-based smartphone threats.

2.1. Threat mitigation using a smartphone firewall

2.1.1. Port-based attack surface mitigation

A *port-based attack surface* is the number of network accessible apps, hosted on the smartphone or on its tethered devices, in terms of ports that are available for a potential attacker to exploit. A smartphone may have a number of network accessible apps, for example RDP port 3389, VNC port 5900, SSH port 22, FTP ports 20 and 21. It is considered best practice to uninstall or disable unnecessary network apps: "*Removing or disabling unnecessary services enhances the security*" (Scarfone et al., 2008). For example, a smartphone may host server-based apps such as Telnet or FTP intended for occasional use. A smartphone user may not wish to install and uninstall these kinds of apps before or after each use. As a consequence, this increases the smartphone's attack surface.

By explicitly configuring the firewall to permit access to intended app ports only, one can significantly reduce the attack surface. Consider the scenario of a remote desktop server app used to manage a smartphones files and photos. Configuring the firewall to permit only RDP traffic destined for port 3389 will reduce the attack surface from a possible 65535 ports to just 1 intended port.

2.1.2. IP-based attack surface mitigation

An *IP-based attack surface* is the number of network accessible apps, hosted on the smartphone or on its tethered devices, in terms of client IP address reachability that are available as a potential attacker threat vector. For example, with respect to smartphone remote management it is recommended to "*Restrict which hosts can be used to remotely administer*" the smartphone where restriction is "*by IP address (not hostname)*" (Scarfone et al., 2008).

Configuration of a smartphone's firewall to comply with best practice recommendations of this kind ensures that IP-based attack surface is significantly reduced. Note, while a smartphone remote management server apps, such as a SSH

or VPN, may provide their own protection in terms of authentication and authorisation, it is considered best practice to also restrict access at the smartphone firewall as part of a defense in depth strategy (Wack et al., 2002).

2.1.3. *IP-based spoof mitigation*

An IP packet's source address may be spoofed (forged) by an attacker in an attempt to trick the smartphone into processing the packet as if it had originated from the smartphone itself or from devices tethered to it. An external attacker may forge IP packets with a set of source IP addresses, for example 192.168.0.0/16, that are associated with the internal private IP network range (IETF, RFC 3330, 2002) but which are inbound on the 3g or WiFi external interface.

A smartphone firewall configured in accordance with standards of best practice will mitigate against the threat of IP spoofing. For example, NIST 800-41rev1 recommendation FBPr1-2 in Table 1 recommends that (spoofed) packets arriving on an external interface claiming to have originated from either of the three RFC1918 (Rekhter et al., 1996) reserved internal IP address ranges should be dropped. This type of attack typically forms part of a Denial of Service Attack (DoS).

2.1.4. *Port scan mitigation*

Port Scanning is a reconnaissance technique that attackers use to determine the network resources of the smartphone and of its tethered devices. Typical TCP-based port scanning involves exploiting the intended use of the TCP protocol by forging TCP header flags.

Firewalls provide an effective way to mitigate against invalid TCP packets. For example, the XMAS TCP port scan where TCP flags FIN, PSH and URG are simultaneously set. In addition to mitigating invalid TCP packets, a firewall that manages TCP communication state are an effective way to mitigate against valid TCP packets that are forged. For example, TCP packets forged to mimic the expected return packets (for example the TCP ACK flag) for outbound TCP traffic requests (for example the TCP SYN flag).

2.1.5. *Tunnel bypass mitigation*

From the point of view of the firewall, the term *tunneling* refers to the practice of encapsulating data from one protocol inside another protocol in order to evade the firewall (Cheswick and Bellovin, 1994). For example, a Skype client typically listens on TCP and UDP port 33033 (Baset and Schulzrinne, 2006).

Table 1 – Extract of NIST-800-41-Rev1: guidelines on firewalls & firewall policy.

ID	Recommendation description	
FBPr1-1	"*deny by default policies should be used for incoming TCP and UDP traffic.*" (Scarfone and Souppaya, 2007).	
	Threat	**Countermeasure**
	No inbound default deny policy	`iptables -P INPUT DROP`
	No outbound default deny policy	`iptables -P OUTPUT DROP`
	No forward default deny policy	`iptables -P FORWARD DROP`
FBPr1-2	"*…an invalid source address for incoming traffic or destination address for outgoing traffic …should be blocked*" that is "*An IPv4 address within the ranges in RFC1918*" and "*An address that is not in an …IANA …range*" (Scarfone and Hoffman, 2009)	
	Threat	**Countermeasure**
	Inbound local 192.168.0.0/16 Src IP Pkt	`iptables -A INPUT -s 192.168.0.0/16 -j DROP`
	Outbound local 192.168.0.0/16 Dst IP Pkt	`iptables -A OUTPUT -d 192.168.0.0/16 -j DROP`
	Inbound forward 192.168.0.0/16 Src IP Pkt	`iptables -A FORWARD -i $iface -s 192.168.0.0/16 -j DROP`
	Outbound forward 192.168.0.0/16 Dst IP Pkt	`iptables -A FORWARD -o $iface -d 192.168.0.0/16 -j DROP`
	Inbound local 10.0.0.0/8 Src IP Pkt	`iptables -A INPUT -s 10.0.0.0/8 -j DROP`
	Outbound local 10.0.0.0/8 Dst IP Pkt	`iptables -A OUTPUT -d 10.0.0.0/8 -j DROP`
	Inbound forward 10.0.0.0/8 Src IP Pkt	`iptables -A FORWARD -i $iface -s 10.0.0.0/8 -j DROP`
	Outbound forward 10.0.0.0/8 Dst IP Pkt	`iptables -A FORWARD -o $iface -d 10.0.0.0/8 -j DROP`
	Inbound local 172.16.0.0/12 Src IP Pkt	`iptables -A INPUT -s 172.16.0.0/12 -j DROP`
	Outbound local 172.16.0.0/12 Dst IP Pkt	`iptables -A OUTPUT -d 172.16.0.0/12 -j DROP`
	Inbound forward 172.16.0.0/12 Src IP Pkt	`iptables -A FORWARD -i $iface -s 172.16.0.0/12 -j DROP`
	Outbound forward 172.16.0.0/12 Dst IP Pkt	`iptables -A FORWARD -o $iface -d 172.16.0.0/12 -j DROP`
FBPr1-3	"*Organizations should also block …IP source routing information*" (Scarfone and Hoffman, 2009)	
	Threat	**Countermeasure**
	SSRR firewall bypass.	`iptables -A FORWARD -m ipv4options –ssrr -j DROP`
	LSRR firewall bypass.	`iptables -A FORWARD -m ipv4options –lsrr -j DROP`
FBPr1-4	"*Organizations should also block …directed broadcast addresses*" (Scarfone and Hoffman, 2009)	
	Threat	**Countermeasure**
	Inbound local directed broadcast	`iptables -A INPUT -d x.x.x.255 -j DROP`
	Outbound local directed broadcast	`iptables -A OUTPUT -d x.x.x.255 -j DROP`
	Inbound forward directed broadcast	`iptables -A FORWARD -i $iface -d x.x.x.255 -j DROP`
	Outbound forward directed broadcast	`iptables -A FORWARD -o $iface -d x.x.x.255 -j DROP`
FBPr1-5	To limit Denial of Service "*a firewall might redirect the connections made to a particular inside address to a slower route if the rate of connections is above a certain threshold.*" (Scarfone and Hoffman, 2009)	
	Threat	**Countermeasure**
	Inbound forward DoS to tethered device	`iptables -A FORWARD -i $iface -d $lanIP -m limit –limit $x/s –limit-burst $y -j ACCEPT`

However, should Skype fail to establish communication over that port, it has the ability to operate on the port required by HTTP (port 80) (Baset and Schulzrinne, 2006; Renals and Jacoby, 2009; Blaich et al., 2008). As a consequence, despite denying traffic for TCP and UDP port 33033, Skype packets may still traverse the firewall unhindered by exploiting the intended purpose of HTTP-based firewall rules.

A smartphone firewall that can perform Deep Packet Inspection (DPI) mitigates the threat of tunnelling. The following is one of many possible Skype signatures used in a *Skype-to-Skype* communication with which a firewall may be configured to filter (Renals and Jacoby, 2009).

^..\x02.....

2.1.6. *Malware traffic mitigation*

A smartphone firewall can be used to mitigate or reduced the flow of Malware communication even in infected phones. Well known Remote Access Trojans (RAT's), such as Androids Geinimi Trojan (Strazzere and Wyatt, 2011), can be blocked in terms of protocol (`TCP`) and ports (`5432`, `4501` and `6543`) from indiscriminately making outbound connections to an external Command and Control (C&C).

RAT's may also communicate with their C&C over HTTP-based ports. For example, DroidDream (https://blog.lookout.com/droiddream/) uses the following URL: http://184.105.XXX.XX:8080/GMServer/GMServlet to transmit IMEI, IMSI and device model information to its C&C server.

In this scenario, as with the *Tunnel Bypass Mitigation* Skype example, a firewall performing DPI with a deny action on outbound HTTP-based packet payloads that contain "`GMServer/GMServlet`" will prevent an infected smartphone communicating with Droidwall's C&C. Note, best practice stipulates the avoidance of once-off firefighting rules where possible and to adopt a default deny rule on outbound traffic (Wack et al., 2002).

3. Smartphone firewall

We define a *smartphone firewall* as a security mechanism that controls traffic flow to and from network-based applications which are hosted by the smartphone itself and/or are hosted by a network of systems tethered to the smartphone in accordance with a security policy.

A *security policy* is a high-level policy document that defines a "*set of rules and practices that specify or regulate how a system or organization provides security services to protect sensitive and critical system resources*" (Shirey, 2000). For example, the restriction of "*user and application access to the built-in web browser, email client, …*" on an enterprise configured smartphone is considered best practice in accordance with NIST-800-124-Revision-1 (Souppaya and Scarfone, 2012b).

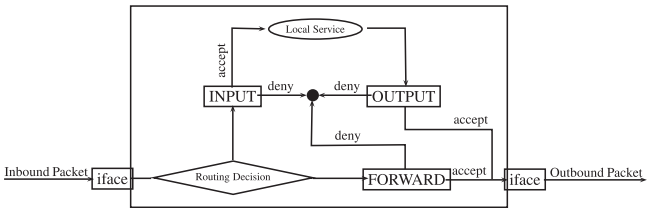


Fig. 1 – Linux iptables filter table chain packet traversal.

A *smartphone firewall configuration* implements a security policy and is defined by a sequence of firewall rules against which all packets traversing the firewall are filtered. The following is an example excerpt of a Linux iptables Android firewall configuration that implements the above NIST-800-124-Revision-1 security policy requirement.

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --match multiport --dports 80,443 -j ACCEPT
iptables -A OUTPUT -p tcp --match multiport --dports 25,110 -j ACCEPT
```

This paper focuses on the Android platform. Android is based upon a modified version of the Linux OS. Therefore, Android uses Linux iptables as its firewall mechanism. An overview of the Linux iptables firewall is outlined in Section 3.1.

3.1. *Linux iptables*

Netfilter (Gheorghe, 2006) is a framework that enables packet filtering, Network Address Translation and packet mangling for Linux. A front-end called *iptables* is used to construct firewall rules that instruct Netfilter how to interpret packets. As a firewall, iptables has stateless, stateful and DPI packet filtering capabilities.

An iptables (firewall, NAT or mangle) rule requires the specification of a *table*, a *chain*, the accompanying *filter conditions* on packet fields that must be matched and an associated *action* outcome. With iptables, there are four tables: `filter`, `nat`, `mangle` and `raw`. A table is a set of chains and it defines the global context for common packet handling functionality. For example, the `filter` table defines the set for firewall rules, while the `nat` table defines the set of rules concerned with Network Address Translation. A chain is a set of rules that define the local context within a table. Rules within a chain are applied to the context defined both by the chain itself and the particular table. This paper focuses on the firewalling aspects of iptables, that is, the `filter` table. There are three built-in chains defined within the `filter` table that govern traffic being routed to (INPUT chain), from (OUTPUT chain) and beyond the firewall itself (FORWARD chain). Fig. 1 illustrates the iptables packet traversal according

to its associated chain. The reader is referred to (Gheorghe, 2006; Suehring and Ziegler, 2006) for further information.

3.1.1. *Example iptables rule syntax*

The following (whitelist) iptables access-control rule states that outbound (`OUTPUT`) TCP packets (`-p tcp`) over the WiFi interface (`-o wifi`) that have originated from the smartphone's Firefox Web browser (`-m owner –uid-owner 10101`) destined to any external Web server (`-d 0.0.0.0/0 –dport 80`) will be permitted (`-j ACCEPT`).

```
iptables -A OUTPUT -o wifi -p tcp -d 0.0.0.0/0
--dport 80 -m owner --uid-owner 10101 -j ACCEPT
```

4. Smartphone firewall configuration management

Management of a smartphone firewall configuration involves either writing low-level command syntax via a Command Line Interface (CLI) or the use of a graphical management console (for example DroidWall (http://code.google.com/p/droidwall/) and WhipserMonitor (http://www.whispersys.com/)). However, firewall management is complex and error prone (Chapple et al., 2009; Wool, 2004). Typical errors range from invalid syntax and incorrect rule ordering, to a failure to uphold a security policy due to lack of GUI-based firewall rule granularity, to errors resulting from the poor comprehension of a firewall configuration (Wack et al., 2002; Marmorstein and Kearns, 2007). An effective firewall configuration may be further hampered by the poor understanding and/or management of the overall high-level smartphone security requirements.

Consider, as a running example, the following security requirements: *permit smartphone browser access to external Web servers* and *apply a default deny policy*. The following is a (naive) iptables firewall configuration that implements these security requirements. Note, for the sake of simplicity only outbound HTTP traffic is considered.

```
iptables -P OUTPUT DROP
iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
```

The above HTTP firewall rule is stateless and stateless firewalls are more prone to attack than a stateful firewall. The lack of authentication in the typical use of network and transport layers means that an attacker can forge TCP packet header attributes that will bypass stateless firewall rules. For example, a forged TCP ack packet can be used to mimic the expected return packets for outbound HTTP traffic requests through the firewall. In comparison, the same attack against a stateful firewall will fail because it will consult both its firewall rules and the current state table before deciding if that inbound TCP packet is to be permitted or denied. It is therefore considered best practice to implement stateful rules (Wack et al., 2002).

```
iptables -A OUTPUT -p tcp --dport 80 -m state --state NEW,ESTABLISHED
-j ACCEPT
```

Note, while the construction of forged TCP packets that have the TCP ack flag set will not open a connection to an application on or behind the smartphone firewall, it is a useful TCP ack scan. Using this type of network scan, it is possible to infer information about the firewall configuration. For example, if the firewall returns a TCP rst packet, the attacker can determine that an internal host exists; if not, it is assumed that the port of the firewall is closed (Lyon, 2008).

It is not enough to just consider stateful firewall rules when implementing the security requirements for Web traffic. There exists a threat of HTTP tunnel bypass (discussed in Section 2.1). To mitigate this kind of threat, one may wish to restrict what applications may communicate over HTTP. The following stateful HTTP firewall rule restricts access to a Web browser only. Note, `$browserUID` is a variable and represents the UID of a particular Web browser running on the smartphone.

```
iptables -A OUTPUT -p tcp --dport 80 -m state --state NEW,ESTABLISHED
-m owner --uid-owner $browserUID -j ACCEPT
```

Furthermore, it may also be prudent to consider the threat of exceeding one's subscribed data plan quota. Service provider excess charges may apply, particularly when the smartphone is in roaming mode. The following builds upon the previous firewall rule to include a 50 MB download capacity on HTTP traffic when communicating over a 3g network.

```
iptables -A OUTPUT -p tcp --dport 80 -m state --state NEW,ESTABLISHED
-m owner --uid-owner $browserUID -m quota --quota 52428800 -o 3g -j
ACCEPT
```

What this running example demonstrates is that deploying firewall rules that upholds the security requirement to permit intended HTTP access is not simply about making only port 80 accessible for all smartphone apps. In practice, generating a smartphone firewall configuration that is aligned with the high-level security policy is challenging, and is largely dependent on the expert-knowledge of the smartphone administrator drawing upon best practice and standards. Section 5 provides a basis with which to model this expert-knowledge in order to automatically manage smartphone firewall configurations on behalf of the non-expert end-user.

5. Security threat model

The security *State* of a smartphone represents attributes of a phone in use that may introduce vulnerabilities and/or influence how threats are mitigated. These attributes may correspond to, for example, user-preferences (indicating for instance, security risk appetite), or how the smartphone is currently used (for instance, a WiFi or 3g Internet connection). While there is potentially a large number of such attributes, for this research we focussed on six which, in-part based on best practice recommendations, have a direct impact on Network Access Controls on smartphones.

5.1. Network interface attribute

A smartphone may be configured to communicate over WiFi and/or 3g networks. Note that a network interface configuration of WiFi and 3g, combined, corresponds to a tethering state. Let *Iface* define the set ($\mathbb{P}\ X$ denotes powerset of X) of possible network interface configurations.

$$Iface \triangleq \mathbb{P}\{\text{wifi}, \text{3g}\}$$

5.2. Network connection attribute

Different network connections may be trusted in different ways. For example, a WiFi connection providing WPA2-Enterprise security may be considered trusted, while an open WiFi connection in a default configuration may be considered untrusted. Let *NetConn* define the possible network connection attribute states.

$$NetConn \triangleq \{\text{trusted}, \text{untrusted}\}$$

5.3. Risk appetite attribute

This user-selected attribute reflects the level of risk that the user is willing to accept (Thinking about risk, 2006). An appetite of `hungry` means that the user is willing to take risks and is satisfied with minimal countermeasures necessary to mitigate threats. An appetite of `averse` means that the user wishes for the most extensive countermeasures, for example, defense in depth.

$$RiskAppetite \triangleq \{\text{averse}, \text{hungry}\}$$

Note, future research may consider additional risk appetite granularity and include `minimalist`, `cautious` and `open` attributes (Thinking about risk, 2006).

5.4. Teleworking attribute

This attribute indicates whether the smartphone is used in teleworking, or non-teleworking mode.

$$Telework \triangleq \{\text{true}, \text{false}\}$$

5.5. Data quota attribute

This user-selected attribute reflects whether the user wishes to apply a maximum data download capacity. If a data quota is to be configured, for example in a scenario where a smartphone is operating in roaming mode, it will applied to a relevant set of white-listed apps.

$$Quota \triangleq \{\text{true}, \text{false}\}$$

5.6. Battery level attribute

The experimental results outlined in Section 8.2 found that the number of firewall rules can have an impact on battery consumption. Therefore, when battery power is low, a user with a low risk appetite may wish to reduce the number of rules in the firewall. Thus, we include the current battery level in the state of the smartphone.

$$Battery \triangleq \{\text{lo}, \text{hi}\}$$

5.7. Security state

The set of all possible states of the smartphone is defined as:

$$State \triangleq Iface \times NetConn \times RiskAppetite \times Telework \times Quota \times Battery$$

5.8. Threats

Let the set *Threat* define the set of all known threats (of interest). A threat is a potential for violation of security (Shirey, 2000) and in this paper we are interested in network-based threats that can be mitigated using a firewall. For example, xmas∈*Threat* represents the threat of a TCP half scan (Lyon, 2008). Let *threatens* define the relationship between threats and states.

$$threatens : Threat \leftrightarrow State$$

where, $threatens(t, s)$ indicates that threat t is considered to threaten a smartphone in state s. For example, we would expect a smartphone in a state with the WiFi interface enabled and an open WiFi connection to be threatened by the `xmas` threat.

5.9. Countermeasures

Let the set *Countermeasure* define the set of known countermeasures. For example, given firewall rule:

$$f_1 = \langle \texttt{iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP} \rangle$$

then $f_1 \in Countermeasure$. In this paper, we are interested in iptables-based countermeasures, and therefore, members of this set are described in terms of iptables command syntax. This could be generalised to the threat ontology described in (Foley and Fitzgerald, 2011) in order to extend to other kinds of countermeasures. Let relation *mitigates* define the threats mitigated by a countermeasure:

$$mitigates : Countermeasure \leftrightarrow Threat$$

where $mitigates(c, t)$ indicates that countermeasure c mitigates threat t. For example, the firewall rule f_1 above mitigates the threat of a TCP half scan, that is, $mitigates(f_1, \texttt{xmas})$.

5.10. Blacklists and whitelists as threats

A blacklist is used to prevent the smartphone from initiating (outgoing) connections to known malicious hosts. Thus, a blacklisted host with IP address A is represented as a threat, denoted $\texttt{blist}_o(A)$, within our model. This threat is mitigated by blocking outgoing packets to A at the smartphone firewall, that is,

$$mitigates(\langle \texttt{iptables -A OUTPUT, ... ,-d } A\texttt{, DROP} \rangle, \texttt{blist}_o(A))$$

A similar interpretation is used for blacklisting inbound (INPUT and FORWARD chains) connections.

A networked Android app has associated port(s), and whitelists are used to define the apps that are permitted to engage in network connections. Whitelists are modelled in terms of threats, whereby a firewall that does not permit a whitelisted app to access the network is treated as a threat and the countermeasure is a corresponding '`ACCEPT`' iptables rule. For example, a whitelisted app that is permitted to initiate outgoing connections on port P is vulnerable to threat denoted $\texttt{wlist}_o(P)$, and we have countermeasure:

$$mitigates(\langle \texttt{iptables -A OUTPUT, ... ,--sport } P\texttt{, ACCEPT} \rangle, \texttt{wlist}_o(P))$$

A similar interpretation is used for whitelisting inbound IP addresses and ports.

5.11. Countermeasure deployment

The countermeasures deployed on a smartphone should mitigate all threats for its current state. We define a deployment operation

$$deploy : State \rightarrow \mathbb{P}\, Countermeasure$$

which selects a suitable set of countermeasures $deploy(s)$ for the state s. The next section describes our current implementation for this operation, however, in general, it should uphold the following property.

$$\forall s : State;\ t : Threats \mid threatens(t, s) \Rightarrow \exists c : Countermeasure \mid c \in deploy(s) \wedge mitigates(c, t)$$

In this paper, the implementation of $deploy(s)$ assumes the correct sequencing of the firewall rules. Future research will consider *structural analysis* techniques (for example (Cuppens et al., 2005)) when automatically generating an anomaly-free firewall configurations.

6. Catalogues of best practice

A best practice standard is a high-level document that defines a set of recommended best practices (countermeasures) to protect sensitive and critical system resources. The following best practice standards NIST 800-41 (Wack et al., 2002), NIST 800-41rev1 (Scarfone and Hoffman, 2009), NIST 800-124 (Jansen and Scarfone, 2008), NIST 800-114 (Scarfone and Souppaya, 2007) and NIST 800-153 (Souppaya and Scarfonea, 2012a) for firewall access control have been encoded within

our model. Excerpts of these catalogues are illustrated in Tables 1–5. For example, Tables 1 and 2 illustrate excerpts of recommended best practice for general firewall configuration (Wack et al., 2002) and firewall configuration whilst teleworking (Scarfone and Souppaya, 2007) respectively.

The advantage of developing catalogues from best practice standards is it provides a basis to automatically generate compliant firewall configurations. For example, NIST 800-41rev1 recommendation FBPr1-2 in Table 1 recommends that (spoofed) packets arriving on an external interface claiming to have originated from either of the three RFC1918 reserved internal IP address ranges should be dropped. Such traffic indicates a denial of service attack typically involving the TCP syn flag. NIST 800-114 recommendation TBP-1 in Table 2 recommends that in a teleworking scenario a firewall should be configured with a whitelist of trusted network-based apps.

Catalogues developed as part of this work extends the catalogues in (Foley and Fitzgerald, 2011) specialised for mobile devices. New best practice catalogues, namely NIST 800-124 (Jansen and Scarfone, 2008), NIST 800-114 (Scarfone and Souppaya, 2007) and NIST 800-153 (Souppaya and Scarfone, 2012a) have also been developed. The catalogue of firewall best practice for smartphones developed as part of this research consists of one hundred and thirty five distinct threat and countermeasure pairs. Future research will extend this catalogue to include knowledge about other best practice standards. Note, the majority of the catalogue countermeasures are templates. For example, the following firewall rule outlined in TBP-2 Table 2 is a template countermeasure that has an UID variable `$appUID` which is modified each time an firewall rule is applied to a locally executing network-based smartphone app.

```
iptables -A OUTPUT -m owner --uid-owner $appUID state
--state NEW,ESTABLISHED,RELATED -j ACCEPT
```

7. Firewall catalogue implementation and deployment

In the smartphone implementation of the catalogue for firewall best practice, we have:

$isMemberOfCategory : Threat \leftrightarrow Category$

where $isMemberOfCategory(t, c)$ indicates that threat category c includes threat t. Table 6 illustrates a fragment of the threat classification developed. The relationship between security states and threats is implemented as:

$threatenState : Category \leftrightarrow State$

where $threatenState(c, s)$ indicates the set of threats categorised within category c threaten the smartphone in state s. The implementation of the *threatens* relation from the model defined in Section 5 is given by the relational composition $isMemberOfCategory \fatsemi threatenState$.

7.1. Threat taxonomy

Having analysed the best practice standards outlined previously, threats where categorised in the following way: *Spoofing*, *Denial of Service*, *Scanning*, *Source Routing*, *Malicious Content*, *Promiscuity Level* and *Non-Audit*. Note, other threat categories could be chosen, for example Microsoft's STRIDE classification (Hernan et al.).

Threats classified as *Spoofing* are those that refer to IP address spoofing. For example, threats described by the FBPr1-2 recommendation in Table 1 are considered spoofing threats.

Denial of Service threats are those that have the capability of flooding network resources. For example, in Table 1 FBPr1-4 recommends IP address broadcast mitigation and FBPr1-5 recommends threshold-limiting to mitigate connection-based denial of service threats. Note, recommendation FBPr1-4 currently considers the more common /24 network broadcast range only and does not consider additional network broadcast ranges for example /25 or /26.

Network information disclosure threats, for example those outlined by NIST 800-114 recommendation TBP-2 in Table 2, are classified as *Scanning* threats.

Source Routing, for example NIST 800-41rev1 recommendation FBPr1-3 in Table 1, is a threat classification where an attacker may specify the route the packet takes through the network and has the potential to bypass firewalls.

From a firewall configuration perspective, *Malicious Content* threats are those that contain malformed application payloads such as URL parameters, form elements and SQL queries. Malicious Content may be mitigated in a variety of ways for example blacklisting known TCP/UDP ports or performing Deep Packet Inspection (DPI) on known malicious signatures. Recommendations TBP-4 in Table 2 and FBP-2 in Table 4 illustrate template DPI firewall rules that mitigates outbound and inbound Malicious Content threat communication.

Threats that are categorised as *Promiscuity Level* are those that refer to IP address (and/or port) reachability in terms of unintended whitelisting or backlisting. That is, an overly-promiscuous firewall configuration (unintended whitelisting), while permitting normal operation of the smartphone app, may expose other apps to unintended threats. Whilst, an overly-restrictive firewall configuration (unintended blacklisting) may prevent normal interoperation of services with the resulting failure of the smartphone app. An example of this is outlined by NIST 800-114 recommendation TBP-1 Table 2.

Non-Audit threats are those that do not log relevant traffic communications. From a compliance perspective, it is considered best practice to log traffic for auditing

Table 2 – Extract of NIST-800-114: User's Guide to Securing External Devices for Telework & Remote Access.

ID	Recommendation description	
TBP-1	Construct an access control whitelist of locally hosted applications trusted for telework network access: "*teleworkers should install and use only trusted software*" (Scarfone and Souppaya, 2007).	
	Threat	**Countermeasure**
	Inbound local application whitelist traffic not permitted	`iptables -A INPUT -m state –state ESTABLISHED,RELATED -j ACCEPT`
	Outbound local application whitelist traffic not permitted	`iptables -A OUTPUT -m owner –uid-owner $appUID state –state NEW,ESTABLISHED, RELATED -j ACCEPT`
TBP-2	…"*silently ignore unsolicited requests sent to it, which essentially hides the device from malicious parties.*" (Scarfone and Souppaya, 2007).	
	Threat	**Countermeasure**
	ICMP ping network scan	`iptables -A INPUT -p icmp -j DROP`
	TCP XMAS network scan	`iptables -A INPUT -p tcp –tcp-flags ALL ALL -j DROP`
	TCP Null network scan	`iptables -A INPUT -p tcp –tcp-flags ALL NONE -j DROP`
	TCP Syn Fin network scan	`iptables -A INPUT -p tcp –tcp-flags SYN,FIN SYN,FIN -j DROP`
	TCP Rst Fin network scan	`iptables -A INPUT -p tcp –tcp-flags FIN,RST FIN,RST -j DROP`
	TCP Port 0 network scan	`iptables -A INPUT -p tcp –dport 0 -j DROP` `iptables -A INPUT -p tcp –sport 0 -j DROP`
TBP-3	"*Use a different brand of Web browser for telework*" (Scarfone and Souppaya, 2007).	
	Threat	**Countermeasure**
	Regular Web browser usage	`iptables -A OUTPUT -p tcp –dport 80 -m owner –uid-owner $untrustedHTTPUID -j DROP`
	Intended telework Web browser usage not permitted	`iptables -A OUTPUT -p tcp –dport 80 -m owner –uid-owner $trustedHTTPUID state –state NEW,ESTABLISHED -j ACCEPT`
TBP-4	"*Configuring primary applications to filter content and stop other activity that is likely to be malicious*" (Scarfone and Souppaya, 2007)	
	Threat	**Countermeasure**
	Outbound local unfiltered traffic	`iptables -A OUTPUT -m -string –algo bm –string '$filterString' -j DROP`
TBP-5	"*Personal firewalls should be configured to log significant events, such as blocked and allowed activity*" (Scarfone and Souppaya, 2007)	
	Threat	**Countermeasure**
	No inbound local audit control	`iptables -A INPUT -j LOG –log-level 7`
	No inbound forward audit control	`iptables -A FORWARD -i $iface -j LOG –log-level 7`

purposes. For example, NIST SP800-114 recommendation TBP-5 in Table 2 outlines teleworking auditing threats and their corresponding firewall mitigation. Similarly, recommendation WiFiBP-2 in Table 3 advocates logging for auditing purposes.

7.2. Security states

The (6-tuple) security *State* space defined in Section 5 provides a total of 64 states in which a smartphone may operate. However, we argue that certain attribute combinations are not valid and therefore the security state space may be reduced to 40. Table 7 illustrates the valid security state matrix.

In this paper, we assume that firewalls under the control of trusted network providers such as a `3g` operator are compliant with best practice standards such as (Wack et al., 2002; Scarfone and Hoffman, 2009). A user with a risk appetite of `hungry`, for example state-7 in Table 7, may therefore not be concerned about threats of IP spoofing, denial of service, port scanning and/or source routing where it is assumed the upstream trusted network provider firewalls are mitigating these kinds of threats.

While the trusted network providers provide firewall mitigation against threats of IP spoofing, denial of service, port scanning and source routing, it is considered best practice to also restrict access at the smartphone firewall as part of a

Table 3 – Extract of NIST-800-153: Guidelines for Securing Wireless Local Area Networks.

ID	Recommendation description	
WiFiBP-1	"*For all their WLAN client devices not authorized for dual connections: Implement the appropriate technical security controls …so that all dual connected configurations are prohibited.*" (Souppaya and Scarfone, 2012a).	
	Threat	**Countermeasure**
	Inbound local spurious iface traffic	`iptables -A INPUT -i $ifaceToDisable -j DROP`
	Outbound local spurious iface traffic	`iptables -A OUTPUT -o $ifaceToDisable -j DROP`
	Inbound forward Spurious iface traffic	`iptables -A FORWARD -i $ifaceToDisable -j DROP`
	Outbound forward Spurious iface traffic	`iptables -A FORWARD -o $ifaceToDisable -j DROP`
WiFiBP-2	Logging: "*often useful for both periodic assessments and continuous monitoring.*" (Souppaya and Scarfone, 2012a).	
	Threat	**Countermeasure**
	No inbound local autit control	`iptables -A INPUT -j LOG –log-level 7`
	No inbound forward autit control	`iptables -A FORWARD -i $iface -j LOG –log-level 7`

Table 4 – Extract of NIST-800-41: Guidelines on Firewalls & Firewall Policy.

ID	Recommendation Description	
FBP-1	Deny "*Inbound or Outbound network traffic containing a source or destination address of 0.0.0.0.*" (Wack et al., 2002).	
	Threat	**Countermeasure**
	Inbound Local 0.0.0.0/8 Src IP Pkt	`iptables -A INPUT -s 0.0.0.0/8 -j DROP`
	Outbound local 0.0.0.0/8 Src IP Pkt	`iptables -A OUTPUT -s 0.0.0.0/8 -j DROP`
	Inbound Forward 0.0.0.0/8 Src IP Pkt	`iptables -A FORWARD -i $iface -s 0.0.0.0/8 -j DROP`
	Outbound forward 0.0.0.0/8 Src IP Pkt	`iptables -A FORWARD -o $iface -s 0.0.0.0/8 -j DROP`
FBP-2	"*Content filtering …virus scanning, filtering, and removal*" (Wack et al., 2002).	
	Threat	**Countermeasure**
	Inbound local unfiltered traffic	`iptables -A INPUT -m -string -algo bm -string '$filterString' -j DROP`
	Inbound forward unfiltered traffic	`iptables -A FORWARD -i $iface -m -string -algo bm -string '$filterString' -j DROP`
FBP-3	Implement stateful rules where possible as "*stateful inspection firewalls are generally considered to be more secure than packet filter firewalls.*" (Wack et al., 2002).	
	Threat	**Countermeasure**
	No whitelist application communication	`iptables -A INPUT -m state -state ESTABLISHED,RELATED -j ACCEPT` `iptables -A OUTPUT -m owner -uid-owner $appUID state -state NEW, ESTABLISHED, RELATED -j ACCEPT`

defense in depth strategy (Scarfone and Hoffman, 2009). As a consequence, security states where the user has risk appetite of `averse` such as state-1, state-3, and state-25 are said to be also threatened by those threats (Table 7).

The number of firewall access-control rules can have an impact on battery consumption (Section 8.2). Therefore, when battery power is `lo`, despite a user having specified a risk appetite of `averse`, the number of firewall rules will be reduced. Security state state-4 is an example, where there is a trade-off of security in depth such as IP spoofing to conserve battery power. Effectively a smartphone with a `lo` battery where a user has specified a risk appetite of `averse` will default to a state where the user is not concerned as much about his/her smartphone's security configuration (risk appetite of `hungry`). For example security states state-15 and state-16.

In contrast, if the smartphone is operating in a state that involves teleworking, for example security states state-1, state-5 and state-10, then a defense in depth strategy will be applied to mitigate all threat categories regardless of the network connection, the risk appetite or the battery level. This is in keeping with NIST 800-114 (Scarfone and Souppaya, 2007) best practice recommendations.

The threats associated with state-29 through to state-40 are the same as those for state-17 through to state-28 except the set of whitelist firewall rules (*Promiscuity Level*) are inclusive of a maximum data download quota that will be specified by the end-user (Section 8).

7.3. *Automatic generation of firewall configurations*

Suitable firewall configurations are automatically generated for each smartphone security state using the information contained in Table 7 and the threat catalogues (for example Table 2). Consider security states state-1 and state-3 where teleworking and non-teleworking occurs. The firewall

Table 5 – Extract of NIST-800-124: guidelines on cell phone and PDA security.

ID	Recommendation description	
CPhBP-1	"*Install and configure additional security controls that are required, including … remote content erasure*" (Jansen and Scarfone, 2008).	
	Threat	**Countermeasure**
	No intended remote erasure whitelist	`iptables -A INPUT -p tcp -sport $port -j ACCEPT` `iptables -A OUTPUT -p tcp -dport $port -j ACCEPT`
CPhBP-2	"*Curb Wireless Interfaces: turn off Bluetooth, Wi-Fi, infrared, and other wireless interfaces until they are needed. "* (Jansen and Scarfone, 2008).	
	Threat	**Countermeasure**
	Inbound local spurious iface traffic	`iptables -A INPUT -i $iface -j DROP`
	Outbound local spurious iface traffic	`iptables -A OUTPUT -o $iface -j DROP`
	Inbound forward spurious iface traffic	`iptables -A FORWARD -i $iface -j DROP`
	Outbound forward spurious iface traffic	`iptables -A FORWARD -o $iface -j DROP`
CPhBP-4	"*Network Access - Malware resident on the device is able to use the device for one or more unauthorized network activities, including port scanning or using the device as a proxy for network communications*" (Jansen and Scarfone, 2008).	
	Threat	**Countermeasure**
	Outbound local malware IP Pkt dropped using default drop as a catch all	`iptables -P OUTPUT DROP`

Table 6 – Extract of threat catalogue.

Detailed threats	Threat category
FBPr1-2 Threats	*Spoofing*
FBPr1-2 Threats	*DoS*
FBPr1-4 Threats	
FBPr1-5 Threats	
TBP-2 Threats	*Scanning*
FBPr1-3 Threats	*Source Routing*
TBP-4 Threats	*Malicious Content*
FBPr1-1 Threats	*Promiscuity Level*
TBP-1 Threats	
TBP-3 Threats	
TBP-5 Threats	*Non-Audit*

configuration generated for security state state-1 will in addition to mitigating threat categories that similarly threaten security state state-3, include audit-based firewall rules in compliance with NIST 800-114 recommendation TBP-5 in Table 2.

While various security states may have been related to the same threat categories, the firewall configuration generated for each security state may be different. Consider security states state-3 and state-25 in Table 7. Both security states are threatened by threats within the category IP spoofing. However, the specific/individual IP spoofing threats such as those described by NIST 800-41rev1 recommendation FBPr1-2 in Table 1 will differ for both security states. Because security state state-25 is concerned with tethering, it must consider additional firewall access-control rules that mitigate IP spoofing threats along its iptables FORWARD chain to protect smartphone tethered devices (Jansen and Scarfone, 2008). Note, in a tethering scenario, the smartphone is an internet gateway for tethered devices.

There are also scenarios where permitted (trusted) network apps in one security state may no longer be permitted in another security state. For example, trusted networked apps such as telnet, FTP or games for example in security state-3 may alternate between whitelists and blacklists in a security state that involve teleworking, for example security state-1. This ensures compliance with NIST 800-114 recommendation TPB-1 in Table 2. That is, only trusted apps defined in accordance with the enterprise-level teleworking security policy may be permitted. Note, while it may be advantageous to deny access to telnet in an enterprise network for a risk appetite of `averse` (for example state-3), it may also be acceptable to restrict access to Telnet for trusted clients (IP address whitelist) while in a home network environment.

In a teleworking scenario, access control is not just defined at the level of IP addresses or TCP/UDP ports, for example prohibiting port 23 (Telnet) or port 80 (HTTP). Access control is also applied at the application level such as UID and Layer-7 filtering. For example, NIST 800-114 recommendation TBP-3 in Table 2 recommends that different Web browsers such as Firefox and Google Chrome, should be used in teleworking and non-teleworking scenario. This is to minimise the Web browser used for general use, which may have become compromised with malicious plugins, from communicating in a teleworking scenario. A set of suitable iptables countermeasures that filter using the owner-match extension — used to match packets based on the identity of the local process that created them — are defined.

While filtering packets using the iptables quota-match extension requires more CPU and memory state and thus will have an impact on battery consumption, we do not relax this security attribute regardless of battery level.

8. MASON

MASON is a prototype automated agent app that manages the smartphone firewall configuration on behalf of the non-expert end-user. Fig. 2 illustrates examples of MASON's Graphical User Interface. The smartphone security state settings interface is illustrated in Fig. 2a where a user may specify the risk appetite, whether or not the smartphone will operate as a tether or in a telework environment, and whether data download quotas should be applied to trusted apps. Fig. 2b, presents the interface which a user may define his/her whitelist and blacklist for (un-) trusted apps. Interfaces for trusted app quota restrictions, server app maximum connection limit and blacklisting by IP addresses are illustrated in Fig. 2c–e respectively.

8.1. *MASON test-bed*

The test-bed used for the prototype was an *Android 2.1, Revision 1* platform on a HTC hero smartphone with an ARMv6 528 MHz processor and a lithium-ion battery with a capacity of 1350 mAh. Note, a rooted and customised Android ROM image that includes additional iptables extensions such as *string match* and *recent match* was used.

8.2. *Firewall configuration and battery consumption correlation*

A number of preliminary experiments where carried out to evaluate the impact of firewall configuration size with respect to battery consumption. The experimental set-up was as follows.

Firewall configurations of 0, 500 and 1000 firewall rules where deployed on the smartphone for each of the three experiments. The battery capacity for each experiment was 100% (fully charged). A 2 GB TCP data-stream was transmitted to the smartphone (from an external machine) where packets are not matched until the last firewall rule in the firewall configuration. Each experiment was repeated 5 times to get the average battery depletion rate. Table 8 illustrates the preliminary findings. The first column reflects the firewall configuration size. The second column reflects the remaining (average) battery level after each experiment. The results indicate that during periods of network communication, the firewall configuration size does have an impact on battery consumption. For example, to filter a 2 GB data-stream, a firewall with a 1000 firewall rules consumed 36% more battery charge than a firewall with 0 firewall rules.

While in practice, smartphones do not tend to process large data-streams and/or be configured with a large number of firewall rules, these experiments were intended to stress test the smartphone. Note, in future work, an additional set of

Table 7 – Matrix of valid security states.

State	Interface	Network connection	Risk appetite	Teleworking	Data quota	Battery	Spoofing	DoS	Scanning	Source routing	Malicious Content	Promiscuity level	Non-audit
state-1	wifi	trusted	averse	true	false	hi	x	x	x		x	x	x
state-2	wifi	trusted	averse	true	false	lo	x	x	x		x	x	x
state-3	wifi	trusted	averse	false	false	hi	x	x	x		x	x	
state-4	wifi	trusted	averse	false	false	lo						x	
state-5	wifi	trusted	hungry	true	false	hi	x	x	x		x	x	x
state-6	wifi	trusted	hungry	true	false	lo	x	x	x		x	x	x
state-7	wifi	trusted	hungry	false	false	hi						x	
state-8	wifi	trusted	hungry	false	false	lo						x	
state-9	wifi	untrusted	averse	true	false	hi	x	x	x		x	x	x
state-10	wifi	untrusted	averse	true	false	lo	x	x	x		x	x	x
state-11	wifi	untrusted	averse	false	false	hi	x	x	x		x	x	x
state-12	wifi	untrusted	averse	false	false	lo	x	x	x		x	x	
state-13	wifi	untrusted	hungry	true	false	hi	x	x	x		x	x	x
state-14	wifi	untrusted	hungry	true	false	lo	x	x	x		x	x	x
state-15	wifi	untrusted	hungry	false	false	hi						x	
state-16	wifi	untrusted	hungry	false	false	lo						x	
state-17	3g	trusted	averse	true	false	hi	x	x	x		x	x	x
state-18	3g	trusted	averse	true	false	lo	x	x	x		x	x	x
state-19	3g	trusted	averse	false	false	hi	x	x	x		x	x	
state-20	3g	trusted	averse	false	false	lo						x	
state-21	3g	trusted	hungry	true	false	hi	x	x	x		x	x	x
state-22	3g	trusted	hungry	true	false	lo	x	x	x		x	x	x
state-23	3g	trusted	hungry	false	false	hi						x	
state-24	3g	trusted	hungry	false	false	lo						x	
state-25	3g,wifi	trusted	averse	false	false	hi	x	x	x	x	x	x	
state-26	3g,wifi	trusted	averse	false	false	lo					x	x	
state-27	3g,wifi	trusted	hungry	false	false	hi						x	
state-28	3g,wifi	trusted	hungry	false	false	lo						x	
state-29	3g	trusted	averse	true	true	hi	x	x	x		x	x	x
state-30	3g	trusted	averse	true	true	lo	x	x	x		x	x	x
state-31	3g	trusted	averse	false	true	hi	x	x	x		x	x	
state-32	3g	trusted	averse	false	true	lo						x	
state-33	3g	trusted	hungry	true	true	lo	x	x	x		x	x	x
state-35	3g	trusted	hungry	false	true	hi						x	
state-36	3g	trusted	hungry	false	true	lo						x	
state-37	3g,wifi	trusted	averse	false	true	hi	x	x	x	x	x	x	
state-38	3g,wifi	trusted	averse	false	true	lo					x	x	
state-39	3g,wifi	trusted	hungry	false	true	hi						x	
state-40	3g,wifi	trusted	hungry	false	true	lo						x	

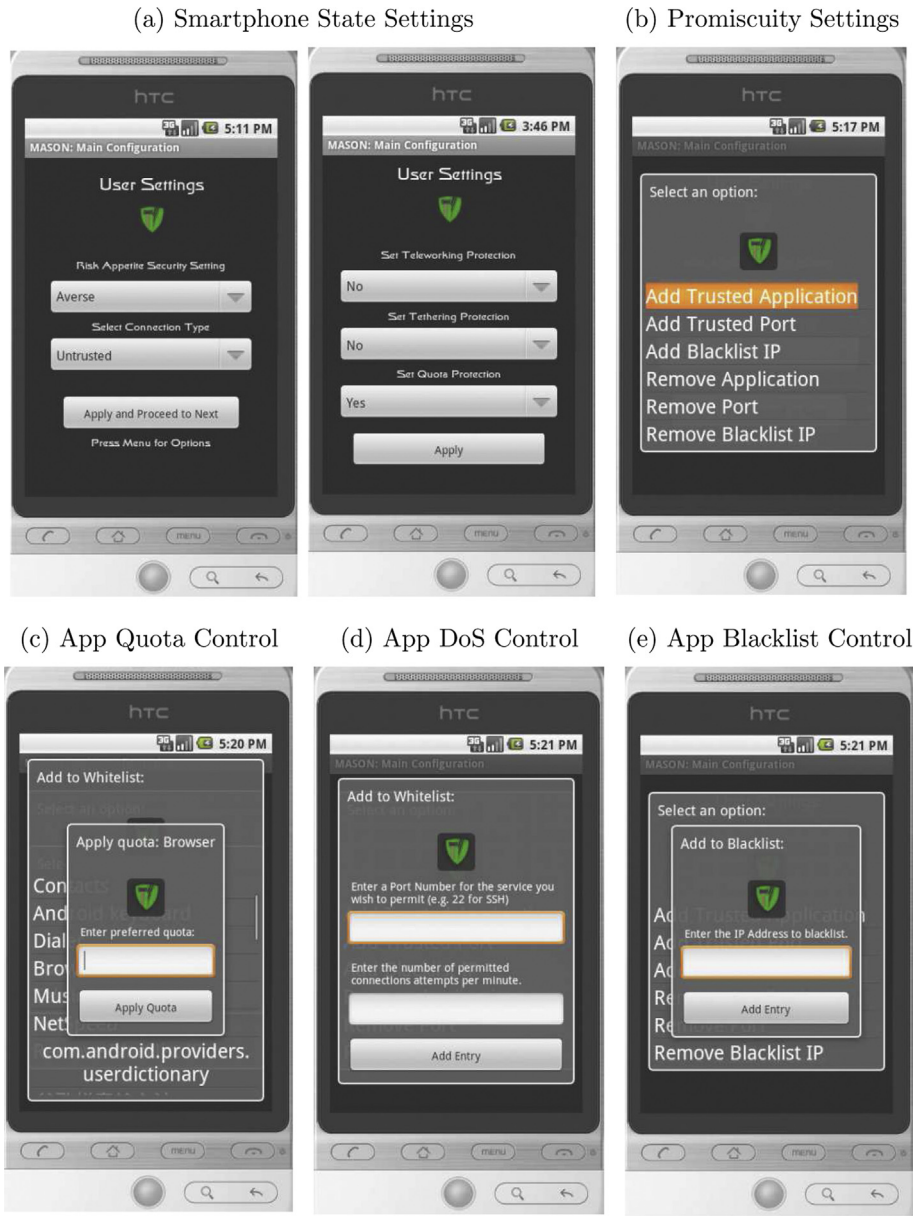


Fig. 2 – Example screenshots of MASON.

experiments that may reflect a more real world scenario will be considered. For example, a 20 MB–90 MB data-stream range (from Web browsing to video streaming (http://www.vodafone.ie/internet-broadband/internet-on-your-mobile/usage/)) tested against firewall configurations consisting of 0, 100 and 250 firewall rules. In addition, experiments where the initial battery capacity is set to 50% and 25% rather than 100% should be considered. As the battery nears a capacity of minimal charge, we conjecture that even a modest sized rule-set consisting a few hundred firewall rules will have a significant impact on battery consumption (Balanza et al., 2005; Saha and Goebel, 2009; Buennemeyer et al., 2007). Therefore, the battery level is considered within the security model presented in Section 5.

Table 8 – Correlation between firewall configuration and battery level.

# of firewall rules	Battery level
0	88%
500	70%
1000	52%

8.3. *MASON discussion*

8.3.1. *MASON extended security states*

The current implementation makes configuration decisions based on six (6) different binary attributes, comprising the

security state. Pruning invalid attribute combinations resulted in 40 configuration scenarios for which corresponding best-practice countermeasures were manually constructed. Exhaustively enumerating state as a means of manually building a catalogue of countermeasures is not scalable; for example, supporting three risk-appetite attribute values, or adding an additional binary attribute potentially doubles the size of the catalogue. We are currently investigating how best practice catalogues can be constructed as a set of constraints over *State*, *Threat* and *Countermeasure*. Given a security state then finding an acceptable configuration of countermeasures correspond to a Constraint Satisfaction Problem (Montanari, 1974) and implemented by the *deploy* operation. Comparable techniques have been successfully used to generate secure configurations (Aziz et al., 2009) given a collection of system constraints.

8.3.2. *MASON conflict-free firewall configuration*

Firewall rules are tested in the sequence in which they appear in the configuration. That is, once a packet has been successfully matched against a firewall rule, no further rule tests are carried out for that packet. Thus, a firewall rule placed out of sequence may unintentionally change the intended meaning of the security configuration and therefore introduce a conflict. That is, it is not possible to consider the semantics of a rule in isolation without also considering that rule in the context of previous rules. For example, consider the following two iptables firewall rules.

```
iptables -I 1 OUTPUT -p tcp --dport 80 -m state --state
NEW,ESTABLISHED -m owner --uid-owner 1000 -j ACCEPT
iptables -I 2 OUTTPUT -p tcp -d 169.254.0.0/16 -j DROP
```

Firewall rule 1 restricts HTTP access to a Web browser only (UID of 1000). Firewall rule 2 (an example of an anti-bogon control) considered in isolation states: *All packets destined to a set of blacklisted hosts are to be denied*. However, the firewall will interpret it to state: *All non-HTTP packets destined to a set of blacklisted hosts are to be denied* based on the semantic relationship rule 2 has with rule 1. Rule 2 should have precedence over rule 1 in this example.

MASON minimises the potential for firewall configuration conflicts as follows. Generalisation firewall rules that apply to app's as a whole, for example anti-port scanning and anti-bogon firewall rules, are given precedence over the disjoint singleton (specific) firewall rules. For the most part, firewall rules are disjoint singleton rules where rule ordering is irrelevant. That is, for each app requiring network access, there is a corresponding firewall rule that also filters based on that app's UID.

The current implementation of MASON assumes that the firewall configuration is conflict free and does not consider structural analysis (Al-Shaer et al., 2005; Cuppens et al., 2005). A future prototype of MASON will consider structural analysis. We conjecture that MASON's extensive use of the iptables UID match filter makes structural analysis on the Android platform feasible and less CPU intensive. That is, rather than having to test a newly added firewall rule against all previous firewall rules for conflicts, one only has to test a new rule against the set of rules that have the same UID match filter. The iptables user-defined chain option (Suehring and Ziegler, 2006) can be used to group firewall rules that have the same UID.

9. Related research

There are a number of existing techniques for static and dynamic analysis of smartphone applications. The authors in (Schmidt et al., 2009) adopt a static analysis approach to detect Android based Malware. In (Egele et al., 2011), a tool called PiOS is developed and uses static analysis techniques to detect data flows in Mach-0 binaries. This provides a basis to detect privacy leaks in Apple's iOS applications. TaintDroid (Enck, 2011) is a smartphone application uses dynamic analysis techniques to detect privacy leaks in Android applications. A machine learning approach is taken in (Shabtai et al., 2012) to detect application anomalies.

There are a number of Android apps for firewall configuration management, for example DroidWall (http://code.google.com/p/droidwall/) and WhipserMonitor (http://www.whispersys.com/). However, the level of access control granularity provided is limited. For example, only egress access control (iptables OUTPUT chain) to whitelist or blacklist apps is considered. The model presented in this paper considers fine-grained ingress (iptables INPUT and FORWARD chains) and egress (iptables OUTPUT and FORWARD chains) access control. In existing works, Android firewall configuration is performed on an ad-hoc basis. For example, there are no recommended guidelines for whitelisting or blacklisting apps in a given security context. In contrast, the automatic generation of smartphone firewall configurations in this research is guided by best practice recommendations.

There are a number of existing techniques that can be used by enterprise security administrators to generate (Foley and Fitzgerald, 2011; Cuppens et al., 2004), query (Foley and Fitzgerald, 2011; Marmorstein and Kearns, 2005) and perform structural analysis (Al-Shaer et al., 2005; Cuppens et al., 2005) on network access control configurations. Future research will explore the effectiveness of these techniques with respect to firewalling on the Android platform.

10. Conclusion

This paper presented a formal model for smartphone security configuration. Catalogues developed as part of this work

extend the catalogues in (Foley and Fitzgerald, 2011) with an emphasis on mobile devices and provided a basis with which to evaluate the security model. MASON may be used by non-expert end-users to automatically generate suitable firewall configurations on the Android platform that are compliant with best practice. Future research will extend the current modelled smartphone firewall catalogues and consider for example catalogues related to smartphone Malware and intrusion detection mitigation. In addition, a future iteration of our (preliminary) security model may consider additional attributes. For example, the physical location of a smartphone where it may be advantageous to prevent a smartphone operating in a teleworking scenario for example when it is located in a certain (untrusted) country or region of the world.

Acknowledgement

The authors would like to thank the anonymous reviewers for their valuable feedback. This research has been supported in part by Science Foundation Ireland grant 08/SRC/11403.

REFERENCES

Al-Shaer ES, Hamed HH, Boutaba R, Hasan M. Conflict classification and analysis of distributed firewall policies. IEEE Journal on Selected Areas in Communications October 2005;23(10):2069–84.

Aziz B, Foley SN, Herbert J, Swart G. Configuring storage-area networks using mandatory security. Journal of Computer Security 2009;17(2):191–210.

Balanza M, Abendan O, Alintanahin K, Dizon J, Caraig B. DroidDreamLight lurks behind legitimate Android apps. In: 6th International conference on malicious and unwanted software (MALWARE) April 2011.

Balanza M, Abendan O, Alintanahin K, Dizon J, Caraig B. Battery discharge characteristics of wireless sensor nodes: an experimental analysis. In: 2nd Conference on sensor and ad hoc communications and networks. IEEE; September 2005.

Baset SA, Schulzrinne H. An analysis of the Skype peer-to-peer internet telephony protocol. In: 25th IEEE international conference on computer communications (INFOCOM) April 2006.

Blaich A, Liao Q, Striegel A, Thain D. Simplifying network management with Lockdown. In: Symposium on usable privacy and security (SOUPS) Pittsburgh, PA, USA July 2008.

Buennemeyer TK, Gora M, Marchany RC, Tront JG. Battery exhaustion attack detection with small handheld mobile computers. In: IEEE international conference on portable information devices, (PORTABLE) May 2007.

Chapple MJ, D'Arcy J, Striegel A. An analysis of firewall rulebase (mis)management practices. Journal of the Information Systems Security Association; February 2009.

Cheswick WR, Bellovin SM. Firewall and internet security: repelling the Wily Hacker. Addison-Wesley; 1994.

Chin E, Felt AP, Greenwood Kate, Wagner D. Analyzing inter-application communication in android. In: 9th International conference on mobile systems, applications, and services, (MobiSys), ACM, USA 2011.

Cuppens F, Cuppens-Boulahia N, García-Alfaro J. Detection and removal of firewall misconfiguration. In: IASTED international conference on communication, network and information security (CNIS) November 2005.

Cuppens F, Cuppens-Boulahia N, Sans T, Miège A. A formal approach to specify and deploy a network security policy. In: 2nd Workshop on formal aspects in security and trust (FAST) August 2004.

Egele M, Kruegel C, Kirda E, Vigna G. PiOS: detecting privacy leaks in iOS applications. In: Network and distributed system security symposium (NDSS), California, USA February 2011.

Enck W. Defending users against smartphone apps: techniques and future directions. In: 7th International conference on information systems security (ICISS), Kolkata, India December 2011.

Evans DL, Bond PJ, Bement AL. Security requirements for cryptographic modules. Federal Information Processing Standards Publication, FIPS; May 2001. p. 140–2.

Felt AP, Finifter M, Chin E, Hanna S, Wagner D. A survey of mobile malware in the wild. In: Proceedings of the 1st ACM workshop on security and privacy in smartphones and mobile devices (SPSM), Chicago, Illinois, USA 2011.

Fitzgerald WM, Neville U, Foley SN. Automated smartphone security configuration. In: 5th International workshop on autonomous and spontaneous security (SETOP), ESORICS, Italy September 2012.

Foley SN, Fitzgerald WM. Management of security policy configuration using a Semantic Threat Graph Approach. Journal of Computer Security (JCS) 2011;19(3). IOS Press.

Gheorghe, 2006 L. Gheorghe, Designing and implementing Linux firewalls with QoS using netfilter, iproute2, NAT and l7-filter, PACKT Publishing, 2006.

Hernan et al. Hernan S, Lambert S, Ostwald T, Shostack A. Uncover security design flaws using the STRIDE approach. http://microsoft.com/.

https://blog.lookout.com/droiddream/.

http://code.google.com/p/droidwall/.

http://trifinite.org/.

http://www.android.com/.

http://www.vodafone.ie/internet-broadband/internet-on-your-mobile/usage/

http://www.whispersys.com/.

Husted N, Saïdi H, Gehani A. Smartphone security limitations: conflicting traditions. In: Proceedings of the 2011 workshop on governance of technology, information, and policies (GTIP), Orlando, Florida 2011.

IETF. RFC 3330: Special-Use IPv4 Addresses. http://ietf.org; 2002.

Jansen W, Scarfone K. Guidelines on cell phone and PDA security: recommendations of the National Institute of Standards and Technology, NIST; 2008. 800–124.

Khadem S. Security issues in smartphones and their effects on the telecom networks, MSc dissertation. Chalmers University of Technology. Sweden: University of Gothenburg; August 2010.

Khan S, Nauman M, Othman AT, Musa S. How secure is your smartphone: an analysis of smartphone security mechanisms. In: International conference on cyber security, cyber warfare and digital forensic, (CyberSec) June 2012.

Landman M. Managing smart phone security risks. In: Information security curriculum development conference (InfoSecCD), Georgia, USA 2010.

Lyon G. NMAP network scanning: official Nmap project guide to network discovery and security scanning. Insecure LLC, CA: United States; 2008.

Marmorstein R, Kearns P. In: Debugging a firewall policy with policy mapping; login: The Usenix Magazine, vol. 32; February 2007. 1, Berkeley, CA, USA.

Marmorstein R, Kearns P. A tool for automated iptables firewall analysis. In: Usenix annual technical conference, Freenix Track April 2005. p. 71–81.

Montanari U. Networks of constraints: fundamental properties and applications to picture processing. Information Science 1974;7:95–132.
Rekhter Y, Moskowitz RG, Karrenberg D, de Groot GJ, Lear E. RFC1918: address allocation for private internets. http://ietf.org; February 1996.
Renals P, Jacoby GN. Blocking Skype through deep packet inspection. In: 42nd International conference on system sciences (HICSS), IEEE Computer Society, Waikoloa, Big Island, Hawaii, USA January 2009.
Ruggiero P, Foote J. Cyber threats to mobile phones, TIP-10-105-01, United States Computer Emergency Readiness Team (US-CERT); April 2010.
Saha B, Goebel K. Modeling Li-ion battery capacity depletion in a particle filtering framework. In: Annual conference of the Prognostics and Health Management Society, San Diego, CA, USA September 2009.
Scarfone K, Hoffman P. Guidelines on firewalls and firewall policy: recommendations of the National Institute of Standards and Technology. NIST Special Publication; September 2009. p. 800–41. Revision 1.
Scarfone K, Souppaya M. User's guide to securing external devices for telework and remote access: recommendations of the National Institute of Standards and Technology, NIST; 2007. 800–114.
Scarfone K, Jansen W, Tracy M. Guide to general server security. NIST Special Publication; July 2008. 800–123.
Schmidt A-D, Bye R, Schmidt H-G, Clausen J, Kiraz O, Yüksel KA, et al. Static analysis of executables for collaborative malware detection on android. In: Proceedings of the 2009 IEEE international conference on communications, ICC'09. Piscataway, NJ, USA: IEEE Press; 2009.
Shabtai A, Fledel Y, Kanonov U, Elovici Y, Dolev S, Glezer C. Google android: a comprehensive security assessment. Security and Privacy, IEEE Computer Society March 2010;8(2).
Shabtai A, Kanonov U, Elovici Y, Glezer C, Weiss Y. "Andromaly": a behavioral malware detection framework for android devices. Journal of Intelligent Information Systems February 2012;38(1).
Shirey R. RFC 2828: Internet security glossary. http://ietf.org; May 2000.
Souppaya M, Scarfone K. Guidelines for securing wireless local area networks (WLANs): recommendations of the National Institute of Standards and Technology, NIST; 2012. 800–153.
Souppaya M, Scarfone K. Guidelines for managing and securing mobile devices in the enterprise (draft): recommendations of the National Institute of Standards and Technology, NIST. Rev 1; 2012. 800–124.
Strazzere T, Wyatt T. Geinimi Trojan Technical Teardown. Lookout Mobile Security June 2011.
Suehring S, Ziegler RL. Linux firewalls. 3rd ed. Novell Publishing; 2006.
Thinking about risk – managing your risk appetite: a practitioner's guide, HM Treasury on behalf of the Controller of Her Majesty's Stationery Office (HMSO); November 2006.
Wack J, Cutler K, Pole J. Guidelines on firewalls and firewall policy: recommendations of the National Institute of Standards and Technology, NIST; 2002. p. 800–41.
Wang Y, Streff K, Raman S. Smartphone security challenges. IEEE Computer Society. Computer December 2012;45(12):52–8.
Wei X, Gomez L, Neamtiu I, Faloutsos M. Malicious android applications in the enterprise: what do they do and how do we fix it? Workshop on secure data management on smartphones and mobiles April 2012. Washington D.C.
Wool, 2004 configuration errors, IEEE Computer 37(6) (2004) 62–67.