



## **MRG Effitas Android AV review**

## Contents

Introduction .....	3
Tests Applied.....	3
Early stage detection.....	3
Detection during installation.....	3
Test Samples.....	4
Security Applications Tested.....	6
Test Results .....	6
Early detection .....	6
Detection during install .....	10

## Introduction

MRG Effitas is an independent IT security research company, with a heavy focus on applied malware analysis. Besides conventional AV efficacy testing and providing samples to other players in the AV field, we regularly test APT detection appliances and enterprise grade IT security products, simulating realistic attack scenarios. In this regard, testing methods have evolved rapidly over the last couple of years as most labs, under the guidance of AMTSO (of which MRG Effitas is a member) strived to conduct “Real World” testing.

## Tests Applied

MRG Effitas performed an in-depth test of several Android AVs. Efficacy of the AV application, the level of protection was measured in real-life scenarios with in-the-wild pieces of malware. This report summarises the results of efficacy tests.

Testing took place on an Android 6.0.0 Genymotion emulator image in April 2018. In cases where ARM native libraries have been used and the AV application could not be installed on an x86 emulator, we opted for a stock Nexus 5x device with Android 6.0.0.

Our efforts were focused on two aspects of the products.

### Early stage detection

Our first scenario focused on an early stage of detection, when test samples have been copied on the SD Card drive of the test device. In the tested scenario, the device has not yet been infected with, malicious APK files have only been downloaded, ready to be installed. In our opinion, a properly designed AV suite should detect threats as early as possible and should not allow users to install potentially dangerous applications on their devices.

Detailed steps were as follows.

1. Having initialised the test device, we installed the AV application and initialised it (accepted EULA, downloaded the latest definition files etc.) When asked, we enabled SD Card scanning features. Due to performance reasons, this option was disabled for most AVs after an out-of-the box initialisation.
2. We set up the application to include the SD Card in the scan scope.
3. We downloaded the sample set to the SD Card and started the scan.
4. We instructed the application to remove all suspicious files.
5. We ran the scan again, until we saw no warning or suspicious files on the device.
6. We collected the remaining samples.

### Detection during installation

The second scenario involved installation of each sample, aiming to check the installation time protection of the AV products.

1. Using adb, we performed an install operation on the device. Following the installation, the AV was informed about the newly installed application, kicking in detection routines.

2. We gave plenty of time for the AV to finish all scanning activities.
3. We created a screen shot of the resulting screen. Should the AV display a warning or an alert, the test was counted as a Pass, no warning resulted in a Miss.
4. Using adb, we uninstalled the sample and went on to test the next one.

Note that on Android, installation of a piece of malware does not necessary mean unwanted consequences for the user, as it is the first launch that kicks in actual malicious code. Having started the sample however, can have detrimental consequences from a security perspective. After the first launch, a piece of malware requesting `SYSTEM_ALERT_WINDOW` permission is able to continuously display a Device Administrator or an Accessibility Admin request screen to the user. In such cases, the user is unable to get rid of the application as they have no access to the launcher, the application drawer or the Settings application to perform an uninstall<sup>1</sup>.

## Test Samples

Testing used an initial 233-sample malware set. All samples have been labelled with one of the following categories.

- **SMS Payment.** The application provides features to send SMS messages to premium rate numbers. Most of the selected samples were able to ‘auto-send’ messages, as they opted for the `SEND_SMS` permission, resulting in a direct financial loss for the victim.
- **Trojan.** Trojans are applications, which display a certain set of features within their description. However, the implemented modules require a wide range of permissions which do not belong to the advertised functionality. A typical example is a flashlight app, which can read the contact list, the GPS position and send them to the Internet.
- **Adware.** The downloaded application implements little or no functionality besides displaying ads on the screen, which, besides legitimate apps, might lure the user into downloading more malware (e.g. with a fake ‘the device is infected! Download this AV now!’ screen). Typical traits of such applications are that they require permissions to draw over other apps for no obvious reason (`SYSTEM_ALERT_WINDOW` permission).
- **Spyware.** We classified a sample Spyware if it leaks information, which can be used to track the user (as most security-conscious users do not wish to be tracked). Ironically, most ad propelled applications qualify as spyware, as they leak IMEI, phone number etc. to the ad provider network.
- **Financial/banking.** This type of malware detects if the user is logged in to a mobile banking session using either a browser or mobile banking application and, for instance, might attempt to display a matching phishing site or to draw an overlay window to fool the user into thinking that the session has ended and that they need to re-authenticate. Typically, such samples use permissions to get the task list, combined with the `SYSTEM_ALERT_WINDOW` permission.

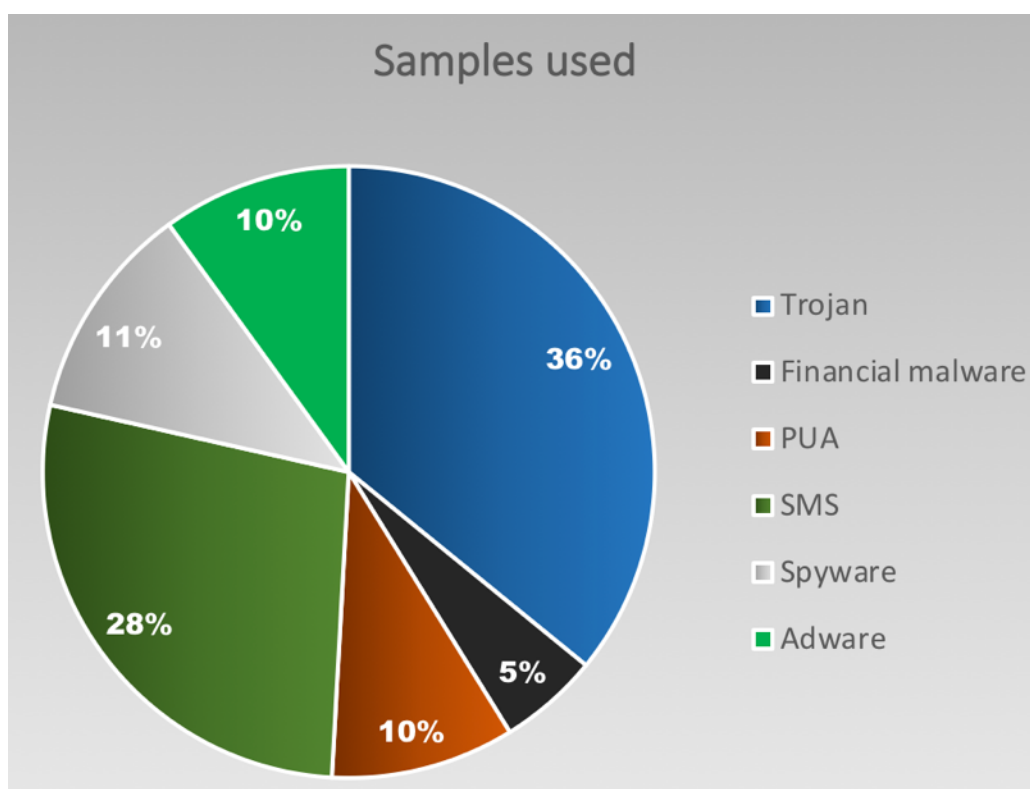
---

<sup>1</sup> Note that in order to mitigate this kind of typical malware behaviour, the Android API design team reviewed the Device Administrator and the Accessibility Admin Request screens to include a checkbox that can be used to prevent the OS from displaying the screen again. This feature however, made its way only to recent revisions of the Android API.

- **PUA.**<sup>2</sup> The term ‘Potentially Unwanted Applications’ denotes applications, which perform actions that are not in alignment with the security-conscious user’s intentions. For instance, applications provided with aggressive advertisement modules usually make it possible for ad campaigners to track individual users, even to assign the device with the user’s demographic properties through social network ad services. Effitas claims that security-conscious users are sensitive regarding their privacy and possibly no application feature can make it up for the users’ private data and browsing habits to be sold over the Internet and a decent AV should let the user know if such an application is about to be installed.

Note that most samples implement several kinds of operation, therefore most samples fall into several categories (for instance, consider a typical piece of malware, which serves malicious ads and if possible, it attempts to obtain the SEND\_SMS permission to send premium rate messages).

Figure 1 depicts the distribution of test samples.



*Figure 1 – Malware sample set distribution*

All samples have been collected in the wild.

<sup>2</sup> Android applications with a social network integrated advertising module often fall into a kind of ‘grey zone’ from a detection perspective, as any application can be turned into a PUA, should the developers include an aggressive advertising module. Hence, we included a chart, which omits PUA samples.

## Security Applications Tested

The following security suites have been selected for testing:

- AVG 6.9.3
- McAfee Security 4.9.5.1944
- Bitdefender 3.3.319
- Symantec Mobile Security 4.1.0.4053
- Kaspersky Internet Security 11.16.4.574
- Lookout Antivirus and Security 10.21.1
- Webroot SecureAnywhere 5.0.0.10759
- Hi Security 4.15.1.1717
- Trend Micro Mobile Security 9.2.2

## Test Results

The tables below show the results of testing under the MRG Effitas Android AV Testing Program.

### Early detection

The following charts show the performance of AV applications when detecting different types of malware from our sample set.



*Figure 2 Overall early detection*

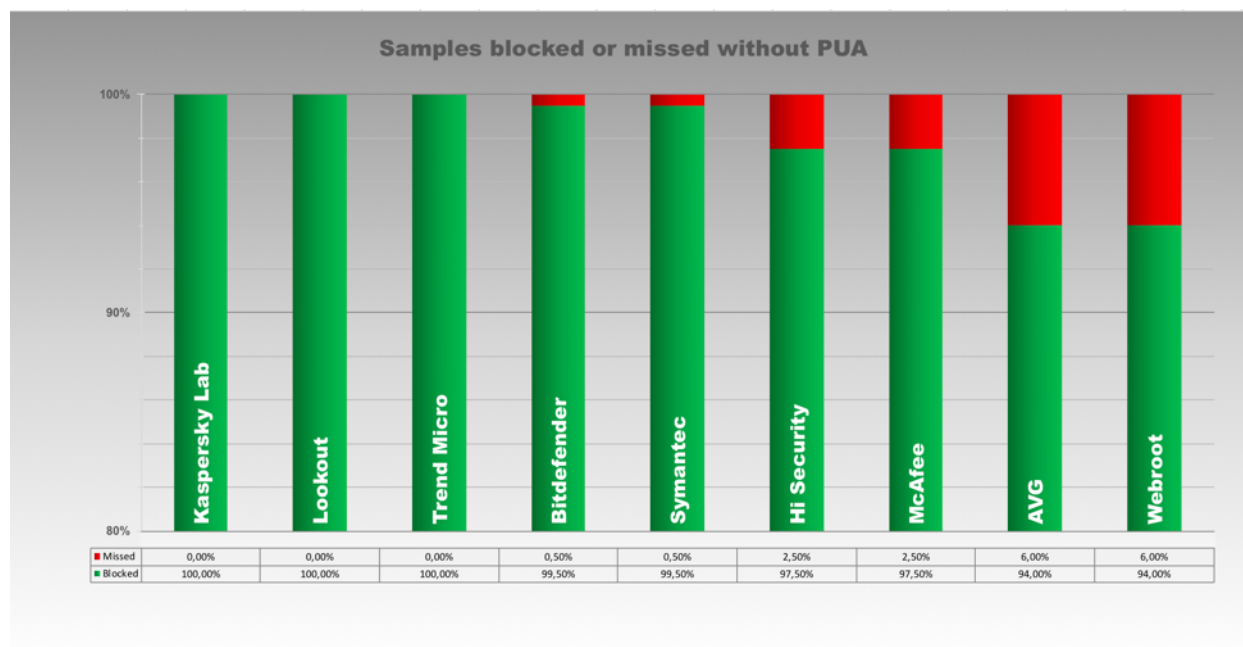


Figure 3 Early detection of non-PUA samples

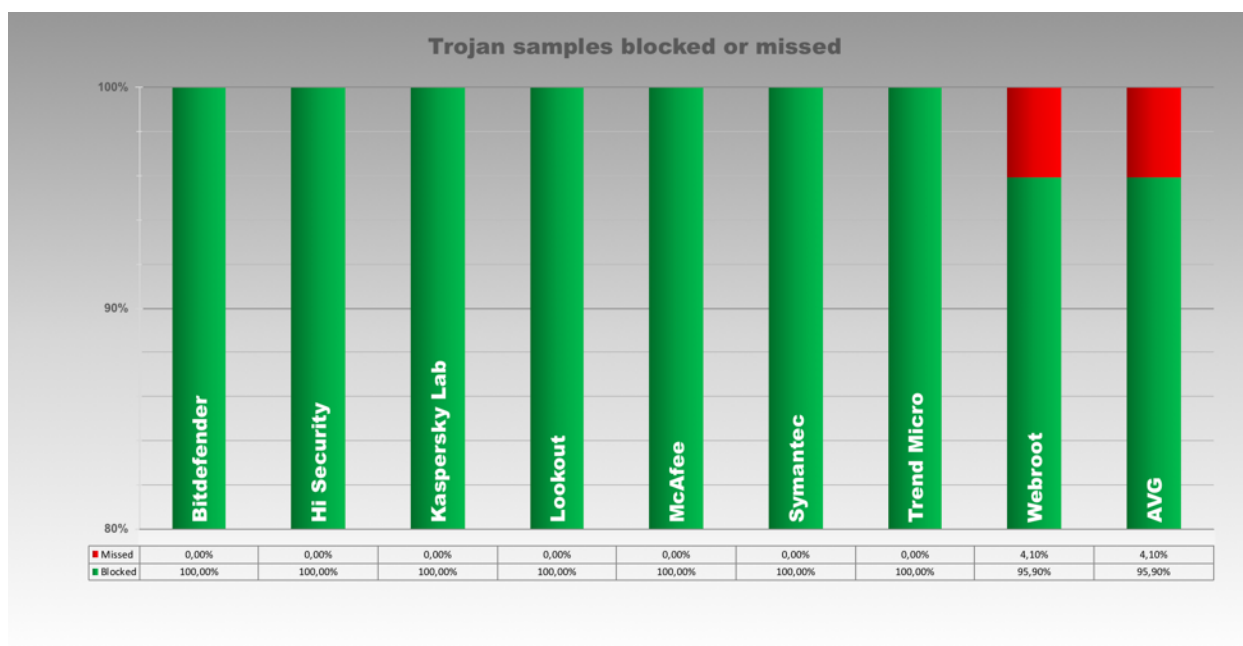
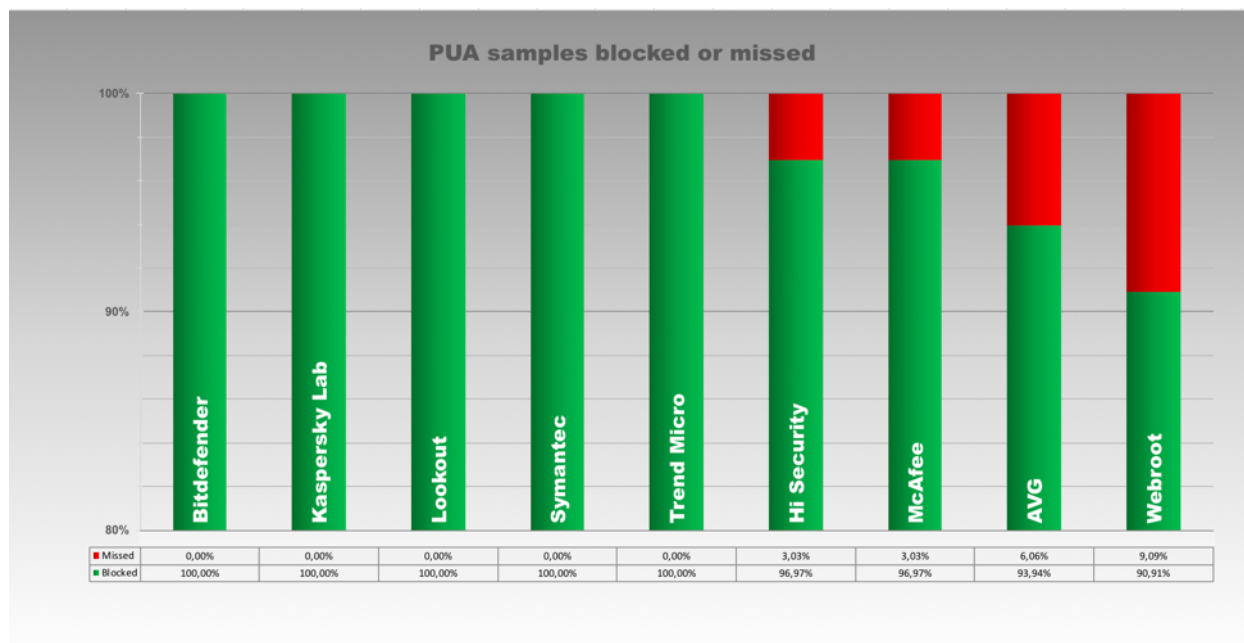
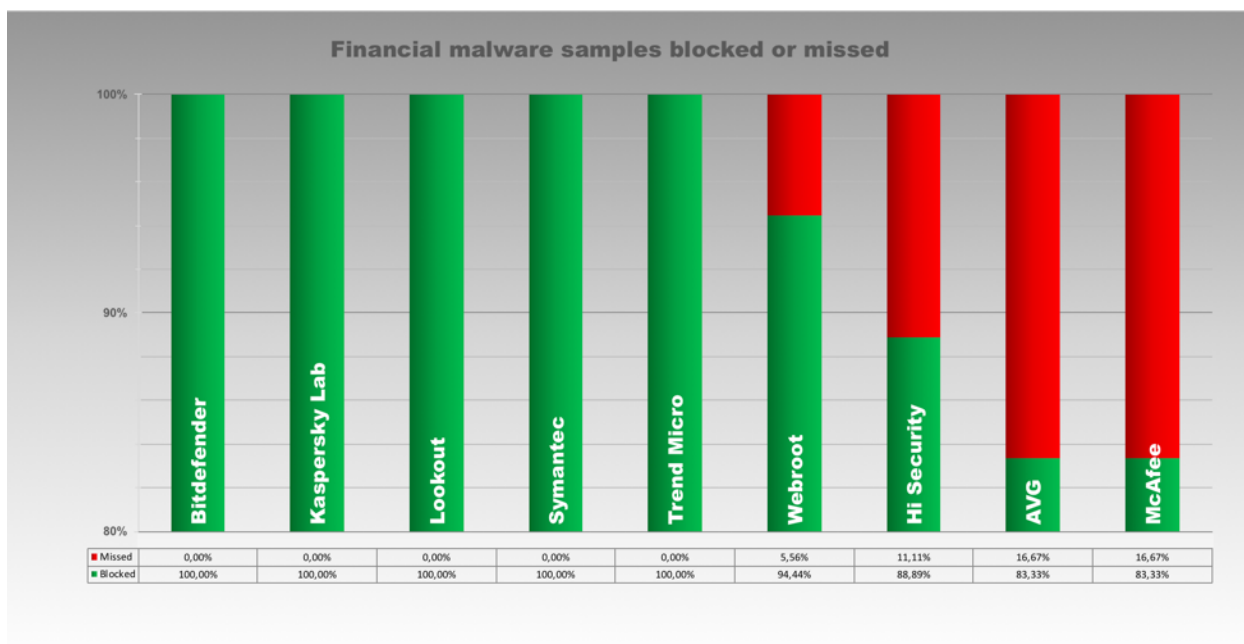


Figure 4 Early detection of trojan samples



*Figure 5 Early detection of PUA samples*



*Figure 6 Early detection of financial samples*



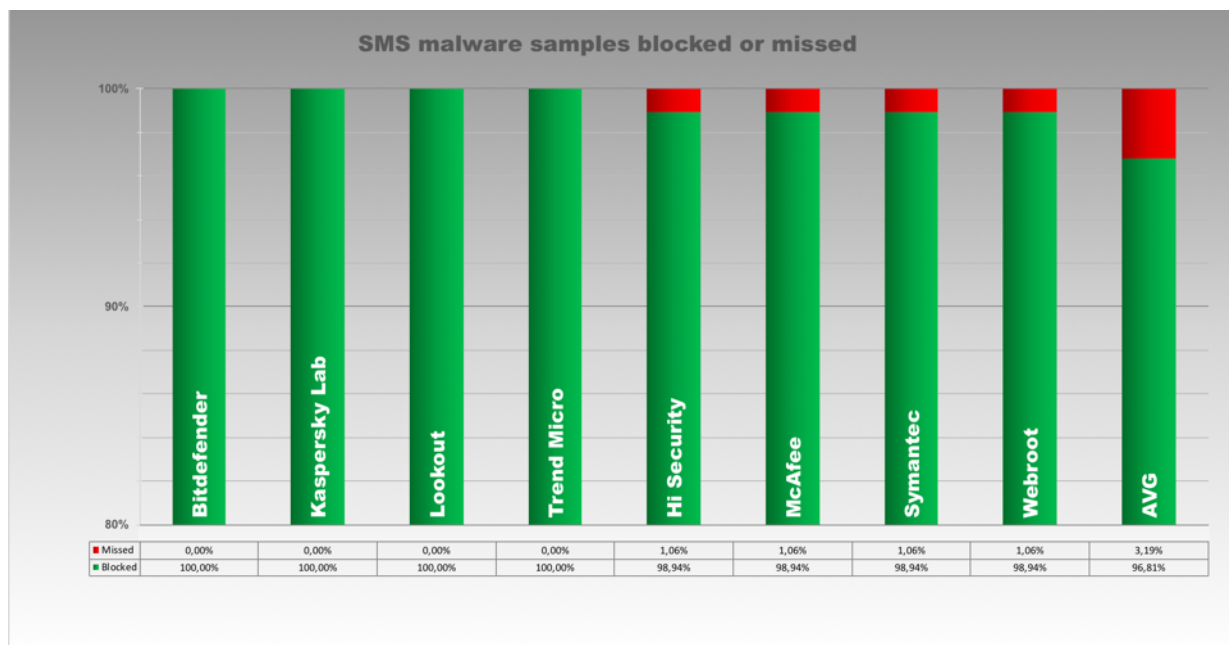


Figure 7 Early detection of SMS fraud related samples

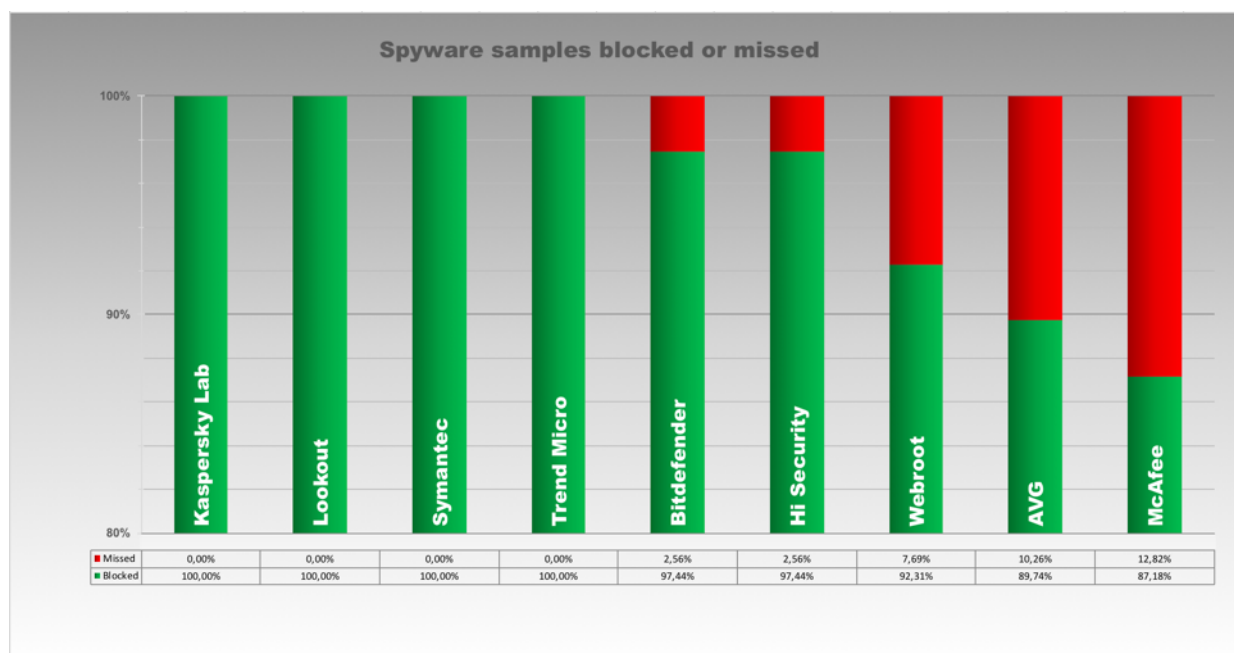


Figure 8 Early detection of spyware

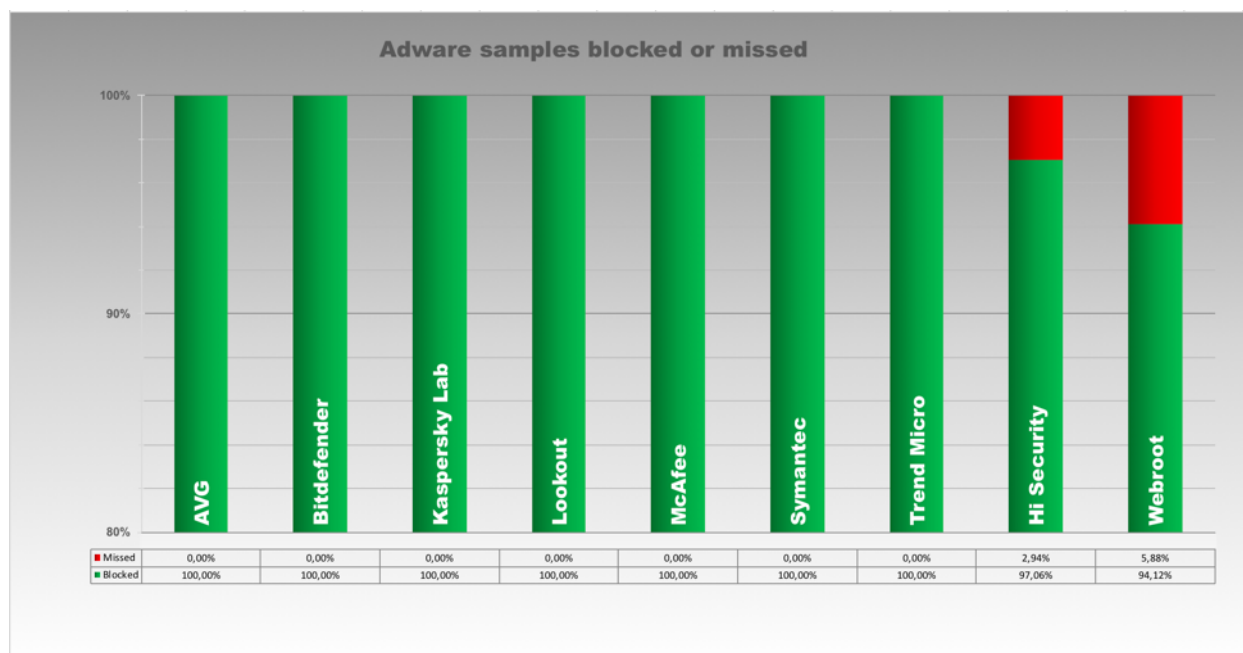


Figure 9 Early detection of adware

## Detection during install

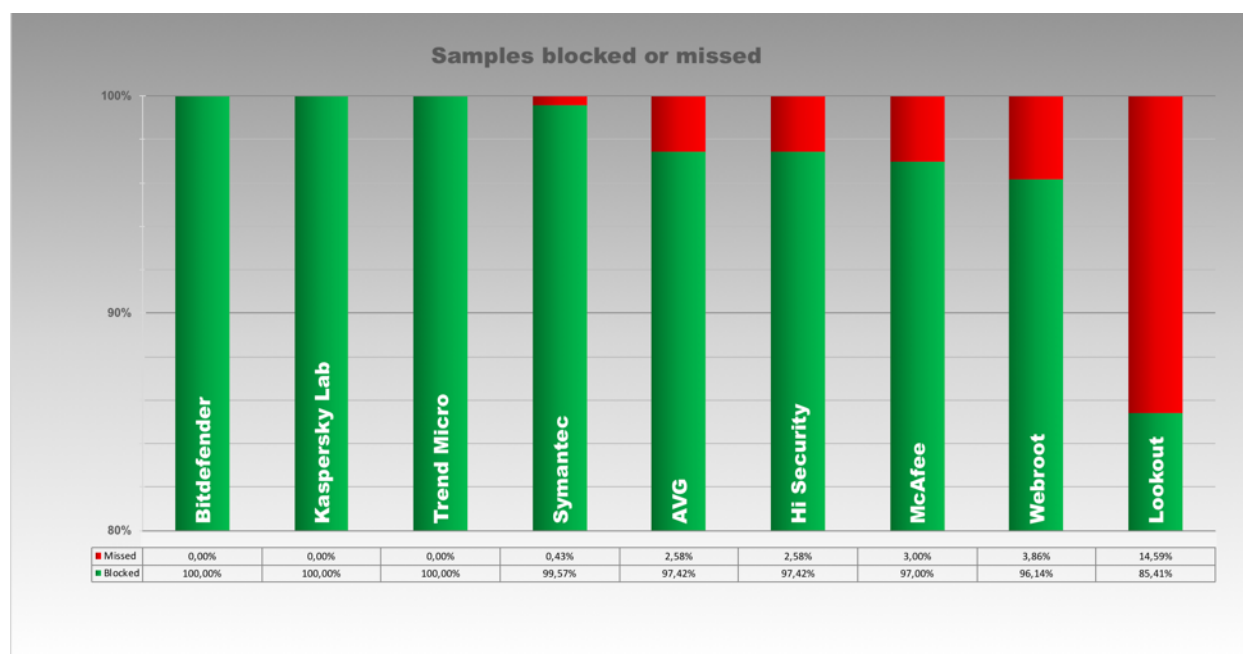


Figure 10 Summary of installation time detection

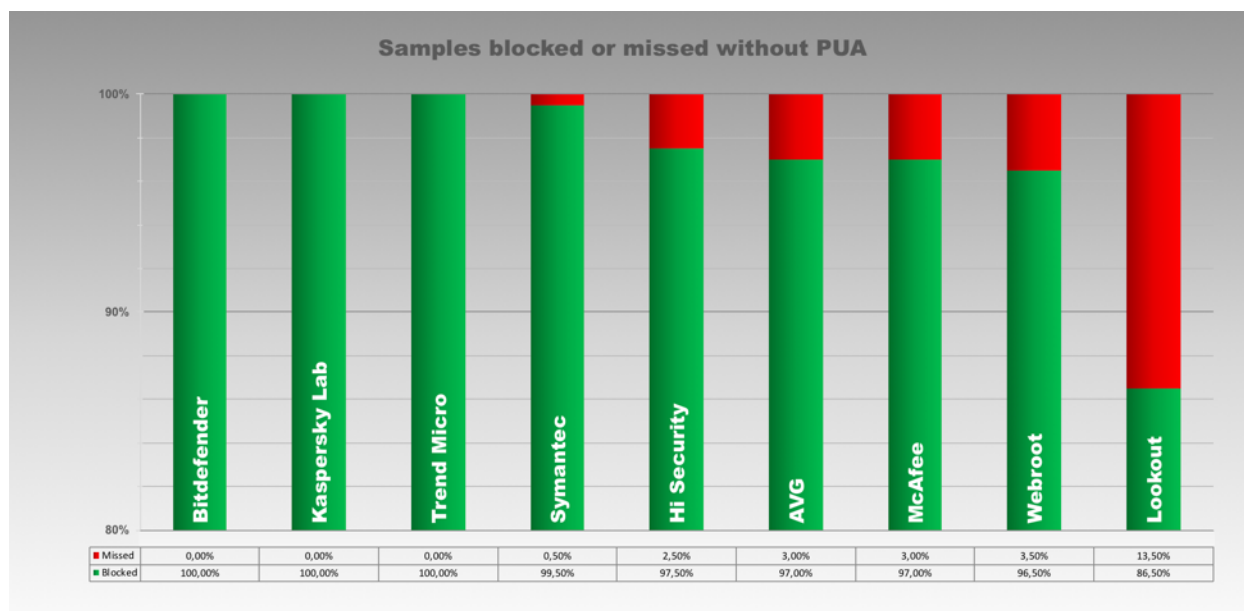


Figure 11 Summary of installation time detection of non-PUA samples

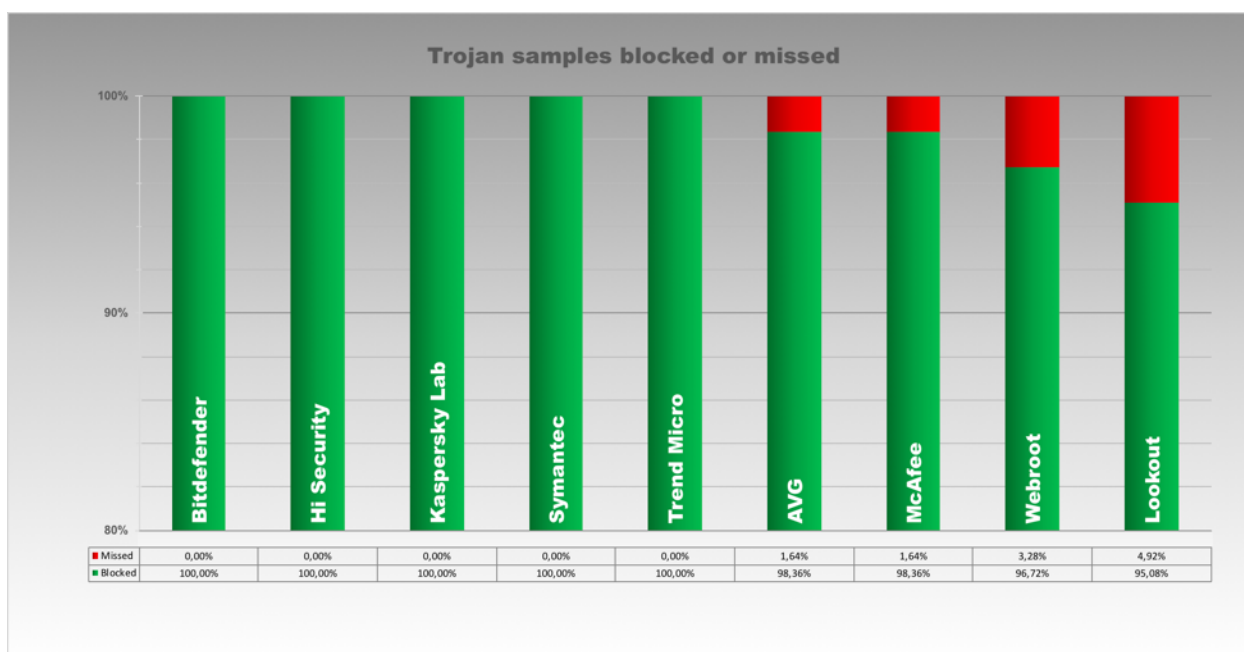


Figure 12 Summary of installation time detection of trojan samples

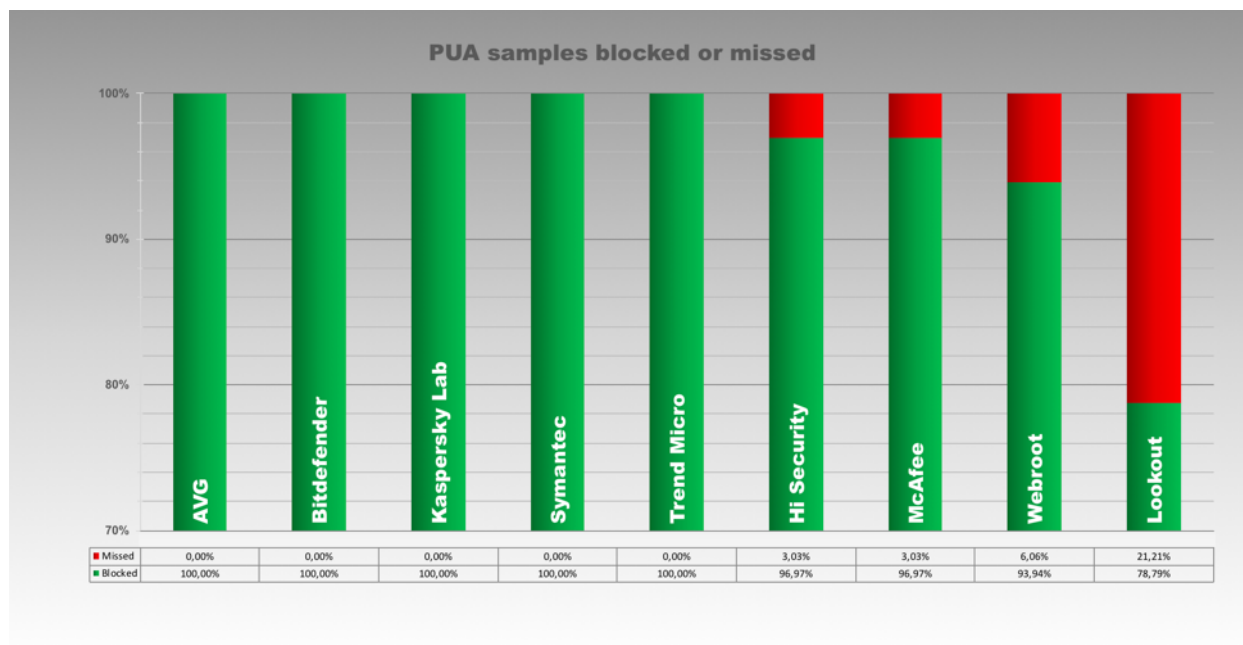


Figure 13 Summary of installation time detection of PUA samples

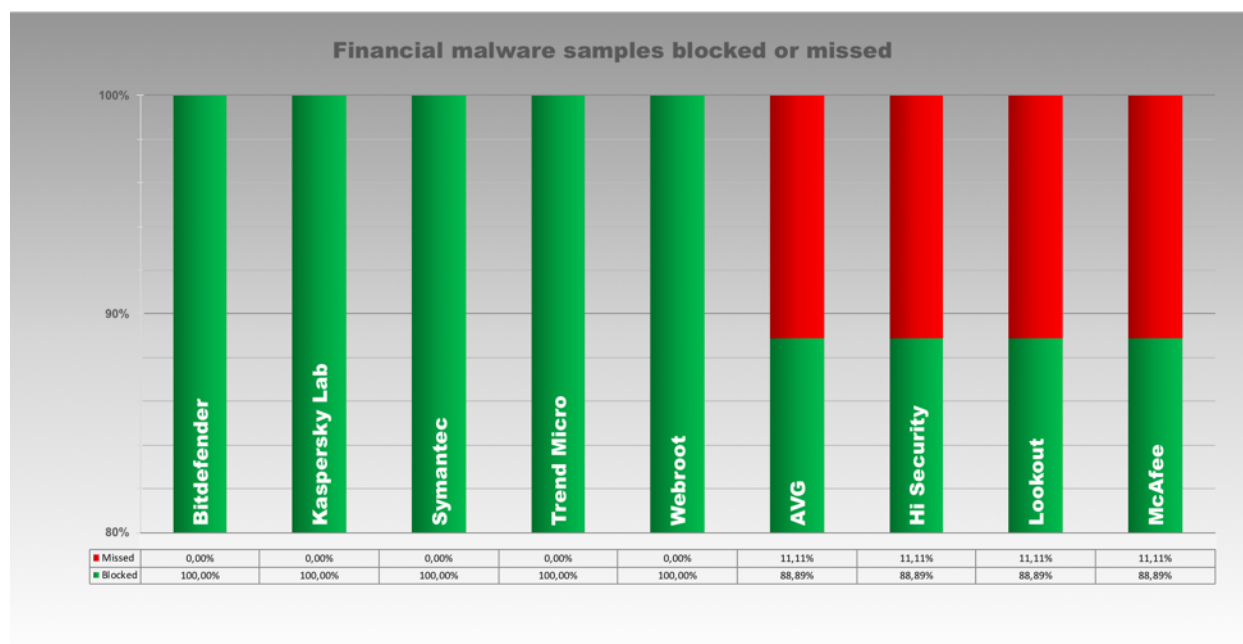


Figure 14 Summary of installation time detection of financial samples

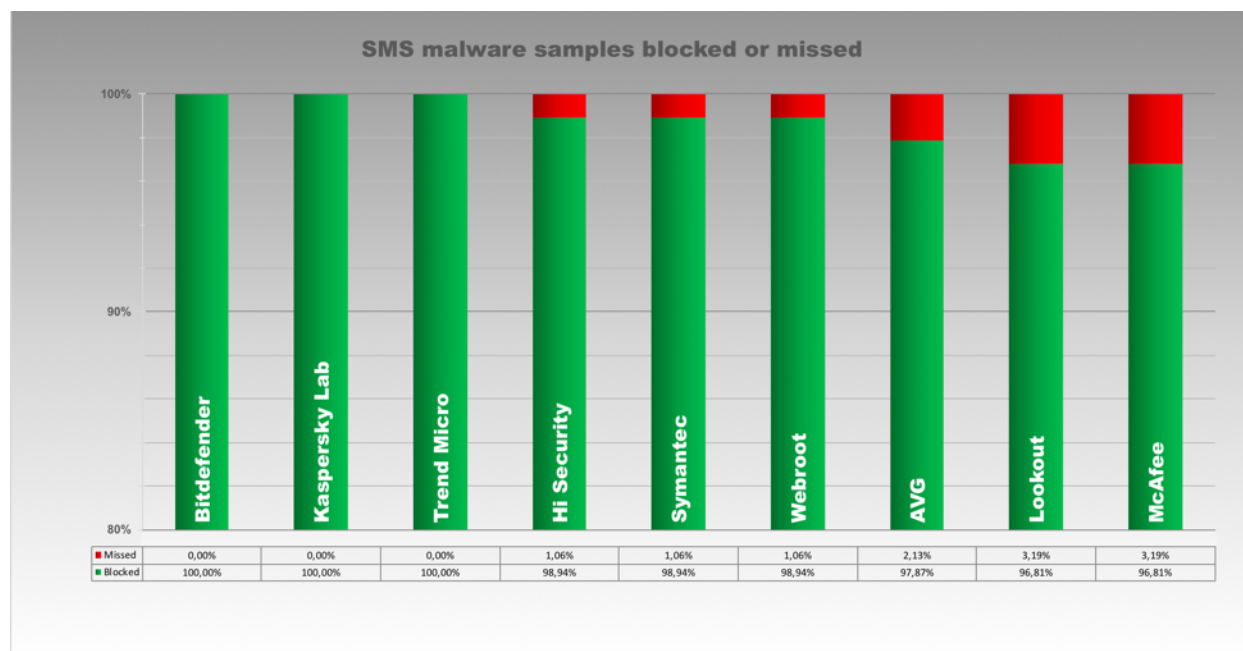


Figure 15 Summary of installation time detection of SMS fraud related samples

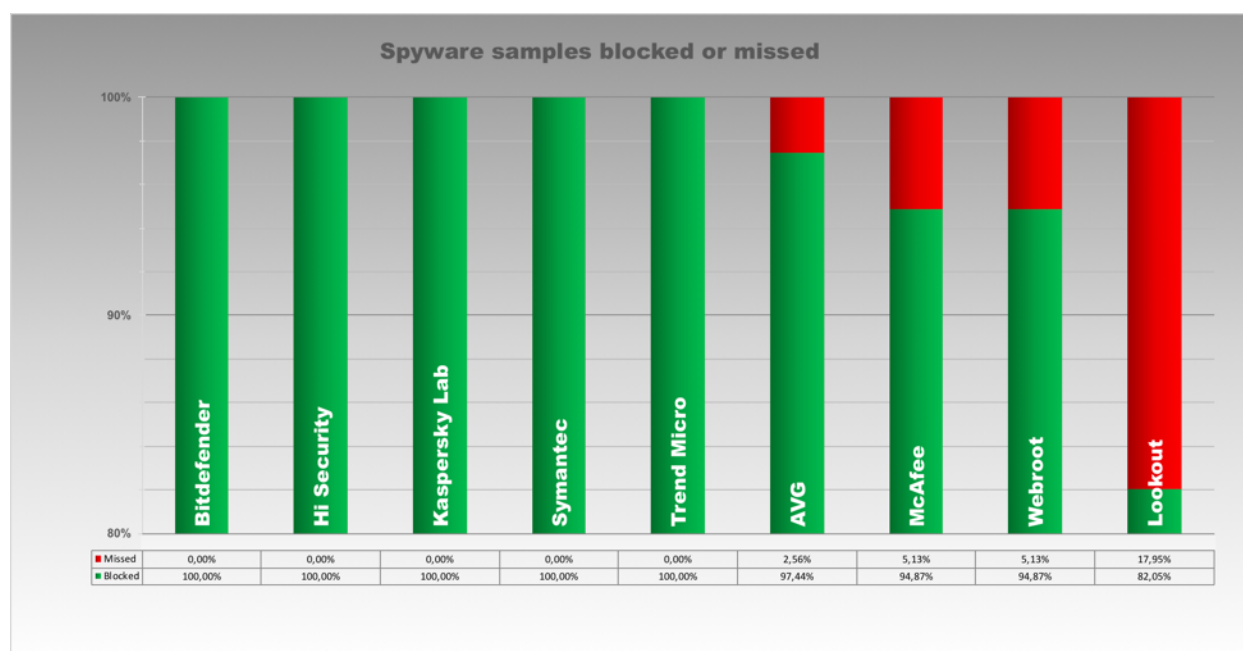


Figure 16 Summary of installation time detection of spyware

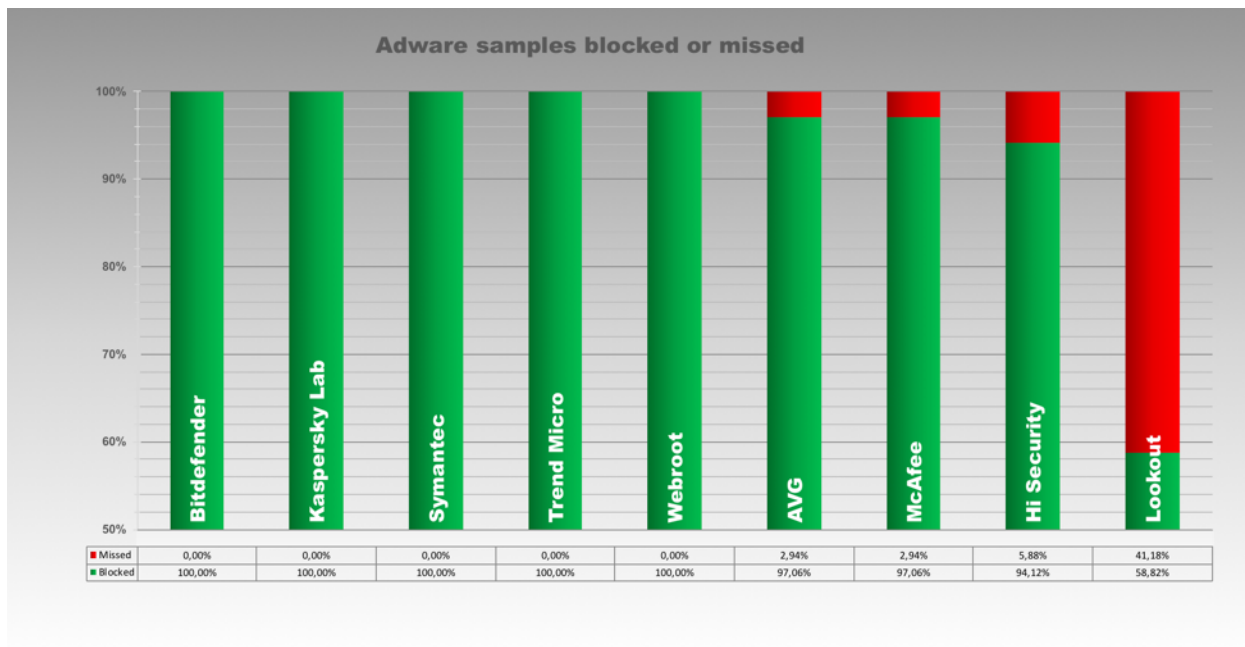


Figure 17 Summary of installation time detection of adware