Online Payments Threats

July 3, 2012

Introduction....2
Tested Products....2
Used Configuration....3
Real Malware Inspiration....3
Total Scores Chart....4
Conclusion....4
About matousec.com....4
Detailed Descriptions of Tests....5
Detailed Results....8

Introduction

We have designed and performed this comparative testing based on the assignment of Kaspersky Lab ZAO. The primary goal of this task was to create a set of tests that would be capable of stealing sensitive information such as users credentials or credit card numbers from selected online payment systems or similar websites; and test various desktop security products in their out-of-box configuration whether they can protect their users against such malicious techniques.

Attacks that result in direct ability of the attacker to steal money from its victims are extremely attractive for today's cyber criminals. These attacks are obviously very dangerous and in fact they are the reason why many users decide to install security products on their computers. Kaspersky Lab ZAO reflected this by introducing a new Safe Money feature in their family of security products. The purpose of the Safe Money feature is to ensure the user's money are safe even if an unknown and thus undetected malware gets into the computer. Many other vendors has also announced their products are ready to protect against financial malware.

Our testing evaluated the real efficiency of the Safe Money feature as well as the quality of other security products.

Further details about the testing including executables and sources of the tests are available to the vendors of the tested products. Please send your requests to research@matousec.com.

Tested Products

We have tested 14 security products. The latest stable version was used for the testing in case of all tested products except for Kaspersky Internet Security, which was tested in version 2013 that was not publicly available at the time of the tests.

The following products were tested:

1. avast! Internet Security 7.0.1426
2. AVG Internet Security 2012.0.2178
3. Avira Internet Security 2012 12.0.0.1085
4. Bitdefender Internet Security 2012 15.0.38.1604
5. ESET Smart Security 5.2.9.1
6. F-Secure Internet Security 2012 12.49.104
7. G Data Internet Security 2013 23.0.0.19
8. Kaspersky Internet Security 2013 13.0.0.3370
9. McAfee Total Protection 2012 11.0.669
10. Microsoft Security Essentials 4.0.1526.0
11. Panda Internet Security 2012 17.01.00
12. Symantec Norton Internet Security 2012 19.7.1.5
13. Trend Micro Titanium Maximum Security 2012 5.2.1035
14. Trusteer Rapport 3.5.1201.76

All the tested products are general purpose anti-virus solutions except for Trusteer Rapport, which is a specialized product designed to protect web communication between service providers and their customers. Its focus is on preventing online fraud committed by financial malware, which perfectly fits to our testing.

Used Configuration

The default, out-of-box, configuration was used with all tested products. Such a selection of products' configurations ensures the fairness of the testing and reflects the actual state of configurations on machines of inexpert users who are most vulnerable to malicious attacks.

We assume our typical user is not able to tweak the settings of the security product. However, the typical user is excepted to perform simple actions that are recommended by the security product. For example, if the product asks the user to perform a computer scan, start the product update, or reboot the computer, the user is expected to be able to click the button and do what he or she is asked to do. The actions we assume the user to do should be initiated by the product itself. We do not assume any activity from the user to improve the security of the machine except the installation of the product and following the product's recommendations.

To be able to test keylogging types of attacks we have disallowed using AutoComplete and similar features of Internet browsers and security products themselves.

All products were installed on 64-bit Windows 7 Service Pack 1. Unless a product recommended the user to use another Internet browser, the default system browser was used, which was 32-bit Internet Explorer.

Each test case was executed separately on a clean machine so that the tests could not affect each other. Unless mentioned otherwise, the tests were started from Windows Explorer and the Internet browser was not running at the moment of the test execution. After the test was started the browser was launched and the user logged in PayPal and eBay.

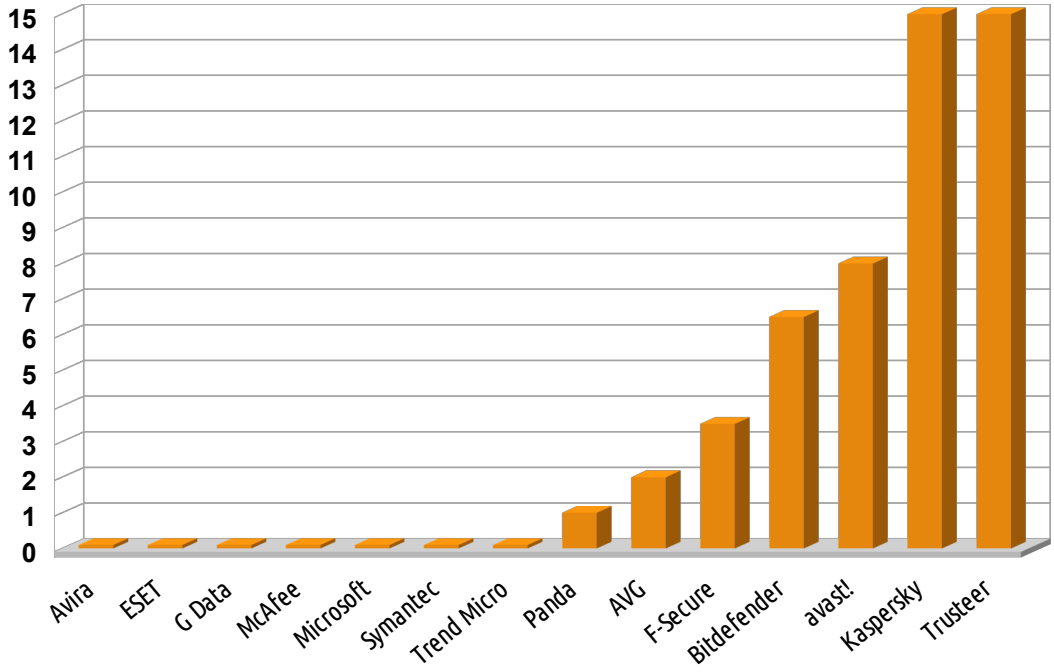
Real Malware Inspiration

We have implemented 15 tests that have a common goal – to steal credentials from PayPal and eBay. Although some parts of the tests may be common, the idea behind each test is different. For detailed descriptions of the tests see Detailed Descriptions of Tests.

The techniques of the created tests have been inspired by real-life malware:

- Zeus trojan horse injects into Internet browsers and steals data submitted in forms. It also implements keylogging techniques to sniff of user passwords and various other techniques to steal passwords. See Wikipedia or SecureWorks for more information about Zeus.
- Sinowal aka Torpig is another well known trojan horse that is able to steal credentials from SSL encrypted communication. See its analysis by Evild3ad.
- Silon malware injects malicious DLL into Internet Explorer in order to steal credentials from POST requests sent to online services. The injection is done by replacing a library that Internet Explorer relies on with a fake malicious library. The ability to steal information is based on hooking various API functions in the Internet Explorer's process. See Trusteer for more information.
- Cidox exploits `AppInit_DLLs` registry entry to inject into Internet Explorer, Mozilla FireFox, Google Chrome, Opera and other browsers. It hooks API functions related to sending data and is thus able to steal any passwords typed into web browsers. More on w4kfu's bl0g.
- Other malware focused on financial cyber crime: SpyEye, Carberp, Yaludle, etc.

Total Scores Chart



For detailed results, see Detailed Results below.

Conclusion

We have tested 14 security products against 15 different techniques that can be used by financial malware to steal credentials to online payment systems. There are plenty of real-life malware samples that do use exactly the same or very similar techniques as the used tests. This is why we would expect most of the tested products to block most of the techniques and keep their users safe from online frauds. However, the results were different.

With the default settings a half of the tested products did not cover a single attack vector. Only four products were able to prevent theft in at least one third of the tested scenarios. Bitdefender Internet Security 2012 and avast! Internet Security were able to block about a half of the tested techniques. Kaspersky Internet Security 2013 and Trusteer Rapport 3.5 successfully blocked all 15 types of attacks.

About matousec.com

matousec.com is a project of Internet & Security Division of AITIS s.r.o., a privately held ICT company from Czech Republic. Since 2006 people behind matousec.com have focused on security of end-user Windows desktops. Our most widely known project is the independent security software testing challenge called Proactive Security Challenge 64.

Besides our public projects, we have established a number of business connections with world-class security software vendors and helped them create the most secure solutions on the market. Over the years our solutions have been implemented into many security products used by millions of users worldwide. Our staff consists of skilled people with a professional approach. We have experts in reverse engineering and low level

and security programming for Microsoft Windows systems, software design and testing, blackbox testing and malware analyzes. We also provide computer security related research and consulting services.

Detailed Descriptions of Tests

Test 01

Test 01 implements a system wide infection. A malicious piece of code is injected into every process in the system using `OpenProcess`, `VirtualAllocEx` and `WriteProcessMemory` calls. Then a remote thread is created inside the target process in order to execute the injected code. This is done using `RtlCreateUserThread` API. In case creation of the remote thread fails the test attempts to hook `NtClose` function inside the target process and redirects it in a way its execution results with the execution of the injected code.

When the injected code is executed it hooks several API functions within the infected process:

- `NtResumeThread` is hooked to enable spreading the infection into child processes. This function is always called when a child process is about to start. By hooking this function it is thus possible to infect the child process before its own code is started. The infection of the malicious code into a child process is done again using `OpenProcess`, `VirtualAllocEx` and `WriteProcessMemory` calls. The execution of the injected code is then ensured by hooking the entry point of the newly created process.
- `LdrLoadDll` is hooked to ensure that all our hooks are installed in the target process. When a process is hooked early during its initialization it may happen that some of the libraries that implement functions we are interested to hook are not loaded at the time of the infection. By hooking `LdrLoadDll` we can install the hooks to the newly loaded libraries immediately after they are loaded.
- `EncryptMessage` and `PR_Write` are hooked in order to sniff on data transferred by Internet browsers over SSL, which is used when sensitive data such as user's credentials are to be sent to an online service. `EncryptMessage` is used by Internet Explorer while the `PR_Write` hook is implemented to support sniffing on traffic from Google Chrome and Mozilla Firefox browsers.

The purpose of this test is to check whether a tested product can prevent a system wide infection of all processes that can lead to infection of Internet browser processes.

Test 02

Test 02 infects the system through the registry key `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders`. After the reboot the critical system process `lsass.exe` loads the malicious DLL and executes its code. Inside `lsass.exe` the malicious code periodically enumerates the running processes and if an uninfected Internet browser process is found it attempts to infect it. The infection is injected into the browser process using `OpenProcess`, `VirtualAllocEx` and `WriteProcessMemory` calls and it is executed using `RtlCreateUserThread` API.

Inside the browser process the malicious code hooks `EncryptMessage` and `PR_Write` functions in order to steal credentials sent to PayPal and eBay.

The purpose of this test is to check whether a tested product can prevent infection of a trusted system process and executing malicious code within its context in order to infect Internet browser processes.

Test 03

Test 03 installs WH_CALLWNDPROCRET hook procedure using `SetWindowsHookEx` API. Because of the limitation

that only 32-bit processes can install Windows hooks to other 32-bit processes and similarly, only 64-bit processes can install Windows hooks to 64-bit processes, the main 64-bit executable of Test 03 firstly starts its 32-bit version and then each of the instances periodically enumerates all process running in the system and waits for the browser processes to appear. When a corresponding browser process is detected, `SetWindowsHookEx` is used to install Windows hook to its threads.

The role of the Windows hook is to inject the malicious DLL to the target Internet browser. Within the browser process `EncryptMessage` and `PR_Write` functions are hooked again.

The purpose of this test is to check whether a tested product can prevent the injection of malicious DLL into browser processes using the `SetWindowsHookEx` method.

Test 04

Test 04 is very similar to Test 03 except that it is the `SetWinEventHook` API that is used to inject the malicious DLL into browser processes.

Test 05

Test 05 installs Browser Helper Object (BHO) in order to infect the Internet browser with malicious DLL. Since Google Chome and Mozilla Firefox do not use the BHO technology, this test works only against Internet Explorer. If a tested product recommends to use a browser that does not support BHO, it automatically passes this test.

The malicious BHO is implemented as a library that is loaded into the Internet Explorer process. As in case of previous test cases, `EncryptMessage` and `PR_Write` functions are hooked to capture the sensitive traffic. The BHO is installed in a way no user's consent is needed for its execution within Internet Explorer.

The purpose of this test is to check whether a tested product can prevent installation of malicious BHOs.

Test 06

Test 06 exploits the infamous `AppInit_DLLs` registry entry in order to inject a malicious library into Internet browser processes. The user credentials are then stolen from within the browser processes using the same hooks as in previous tests cases.

Test 07

Test 07 implements the binary planting method in order to inject a malicious version of `version.dll` library into the Internet browser. The fake library is copied to the installation directory of the Internet browser. When the browser is started it loads `version.dll` to its process. Its installation directory is searched first for this library before the system directory is searched. This is why the fake version of the library is loaded instead of the original one.

The fake `version.dll` implements all functions as the original `version.dll` by loading the original library into the browser process and redirecting all the exports to the original functions. The original functionality is thus preserved and no problems occur due to the installation of the fake library.

The fake library code hooks `EncryptMessage` and `PR_Write` functions to capture the sensitive traffic.

The purpose of this test is to check whether a tested product can prevent the binary planting method of DLL injection.

Test 08

Test 08 is the first keylogging test case. It uses `SetWindowsHookEx` API to install a system wide WH_KEYBOARD hook. After the hook is installed, applications that accept keyboard input through WM_KEYDOWN and WM_KEYUP messages load a malicious DLL into their processes and call its callback routine every time they receive the mentioned messages. The malicious code then extracts the information about which key was pressed from these messages. This is how Test 08 steals the credentials.

As in cases of Test 03 and Test 04, in order to support both 32-bit and 64-bit applications, there are two Test 08 executables and the primary 64-bit program runs the 32-bit version immediately after it is started.

The purpose of this test is the same as the one of Test 03 except that Test 08 uses the keylogging method to steal the credentials and should thus be able to bypass a protection that is based on preventing `EncryptMessage` or `PR_Write` hooks.

Test 09

Test 09 implements the `GetKeyState` keylogging method. This technique does not rely on any kind of code injection and just sniffs on the keystrokes typed by the user.

The purpose of this test is to check whether a product is able to protect the credentials against the `GetKeyState` keylogging method.

Test 10

Test 10 implements the `GetAsyncKeyState` keylogging method and is very similar to Test 09.

Test 11

Test 11 is a keylogger based on WH_KEYBOARD_LL hook installed by `SetWindowsHookEx` API. This special kind of hook does not rely on a DLL injection. The messages with the pressed keys are delivered directly to the malicious application.

The purpose of this test is to check whether a product is able to protect the credentials against the WH_KEYBOARD_LL hook keylogging method.

Test 12

Test 12 does the same thing as Test 11 except that the WH_JOURNALRECORD hook is used instead of the WH_KEYBOARD_LL hook.

Test 13

Test 13 implements another keylogging technique that is based on attaching the input of the test's thread to the input of the foreground window thread. It periodically attempts to find a foreground window using `GetForegroundWindow` API and then attach its input to the thread that owns the foreground window using `AttachThreadInput` API. After the input is attached, `GetKeyboardState` is used to obtain the information about the input itself.

The purpose of this test is to check whether a product is able to protect the credentials against the mentioned keylogging method.

Test 14

Test 14 implements the raw input device keylogging method. It uses `RegisterRawInputDevices` API to register its own raw input device that receives messages about all keyboard input in the system. When there is an input ready in the system the window associated with the raw input device receives WM_INPUT message and using `GetRawInputData` API it obtains information about the input itself.

The purpose of this test is to check whether a product is able to protect the credentials against the raw input device keylogging method.

Test 15

Test 15 implements the Direct Input 7 keylogging method. The Direct Input 7 interface is used to create a SysKeyboard device, which is then pooled in a loop using `GetDeviceState` method. On the output this method returns the state of the keyboard from which the pressed keys are extracted.

The purpose of this test is to check whether a product is able to protect the credentials against the Direct Input 7 keylogging method.

Detailed Results

The result of each test can either be PASSED, FAILED, or OTHER:

- PASSED means that the product prevented stealing credentials from both PayPal and eBay. A product receives 1 point for this result.
- FAILED means that the product failed to prevent stealing credentials either from PayPal or eBay or both. A product receives 0 points for this result.
- OTHER means that a problem such as a system crash occurred during the test. A product receives 0.5 points for this result.

Every result may be supported by a comment that explains the details of the particular case.

avast! Internet Security 7.0.1426

Total Score: 8/15

This product recommended to install Google Chrome as the default browser. This product implemented a sandbox feature in which all unknown processes were executed. By default this sandbox was very restrictive and terminated every program after about 15 seconds (probably) if it did not present a behavior expected by the product, which was probably some kind of interaction with the user. This was why we had to start the tests from a sandboxed console, which allowed us to run tests inside the sandbox for more than 15 seconds.

Test Case	Result	Comment
Test 01	PASSED	The test's process was forcefully terminated.
Test 02	PASSED	The registry was not modified outside the sandbox.
Test 03	FAILED	
Test 04	PASSED	The sandboxed process was not able to infect the browser process running outside the sandbox.

Test 05	PASSED	Google Chrome does not use BHOs.
Test 06	PASSED	The registry was not modified outside the sandbox.
Test 07	PASSED	The malicious files were installed within the sandbox only.
Test 08	PASSED	The browser outside the sandbox was not infected.
Test 09	FAILED	
Test 10	FAILED	
Test 11	FAILED	
Test 12	FAILED	
Test 13	FAILED	
Test 14	PASSED	The keystrokes from the outside of the sandbox were not captured.
Test 15	FAILED	

AVG Internet Security 2012.0.2178

Total Score: 2/15

Test Case	Result	Comment
Test 01	PASSED	A threat was detected, moved to the virus vault, and it was recommended to reboot the computer immediately, which was sufficient. If the computer was not rebooted the processes in the system would remain infected with the malicious DLL.
Test 02	FAILED	
Test 03	FAILED	
Test 04	FAILED	
Test 05	PASSED	A threat was detected, moved to the virus vault, and it was recommended to reboot the computer immediately, which was sufficient because the BHO registry entries were also removed and so was the malicious DLL from the disk.
Test 06	FAILED	
Test 07	FAILED	
Test 08	FAILED	
Test 09	FAILED	
Test 10	FAILED	
Test 11	FAILED	
Test 12	FAILED	
Test 13	FAILED	

Test 14	FAILED	
Test 15	FAILED	

Avira Internet Security 2012 12.0.0.1085

Total Score: 0/15

Test Case	Result	Comment
Test 01	FAILED	
Test 02	FAILED	
Test 03	FAILED	
Test 04	FAILED	
Test 05	FAILED	
Test 06	FAILED	
Test 07	FAILED	
Test 08	FAILED	
Test 09	FAILED	
Test 10	FAILED	
Test 11	FAILED	
Test 12	FAILED	
Test 13	FAILED	
Test 14	FAILED	
Test 15	FAILED	

Bitdefender Internet Security 2012 15.0.38.1604

Total Score: 6.5/15

Test Case	Result	Comment
Test 01	OTHER	The test caused BSOD probably due to some kind of protection of Bitdefender that was not compatible with the technique of this test.
Test 02	FAILED	Although the test was detected and blocked by Active Virus Control as potentially dangerous, the infection was not removed

		from the registry and the test successfully infected lsass.exe and the browser after the reboot.
Test 03	FAILED	
Test 04	FAILED	
Test 05	FAILED	Although the test was detected and blocked by Active Virus Control, the BHO was not uninstalled and even after the reboot the credentials could be stolen.
Test 06	FAILED	Although the test was detected and blocked by Active Virus Control, the registry infection was not removed and even after the reboot the credentials could be stolen.
Test 07	FAILED	
Test 08	FAILED	
Test 09	PASSED	The test was successfully detected and blocked by Active Virus Control.
Test 10	PASSED	The test was successfully detected and blocked by Active Virus Control.
Test 11	PASSED	The test was successfully detected and blocked by Active Virus Control.
Test 12	PASSED	The test was successfully detected and blocked by Active Virus Control.
Test 13	PASSED	The test was successfully detected and blocked by Active Virus Control.
Test 14	PASSED	The test was successfully detected and blocked by Active Virus Control.
Test 15	FAILED	

ESET Smart Security 5.2.9.1

Total Score: 0/15

Test Case	Result	Comment
Test 01	FAILED	
Test 02	FAILED	
Test 03	FAILED	
Test 04	FAILED	
Test 05	FAILED	
Test 06	FAILED	

Test 07	FAILED	
Test 08	FAILED	
Test 09	FAILED	
Test 10	FAILED	
Test 11	FAILED	
Test 12	FAILED	
Test 13	FAILED	
Test 14	FAILED	
Test 15	FAILED	

F-Secure Internet Security 2012 12.49.104

Total Score: 3.5/15

Test Case	Result	Comment
Test 01	FAILED	
Test 02	FAILED	
Test 03	OTHER	The product warned about a potentially harmful application in its System Modification Attempt dialog, but even if the user's answer was to block the threat, the credentials could be stolen. However, it was not possible to execute the offending test again.
Test 04	FAILED	
Test 05	FAILED	
Test 06	FAILED	
Test 07	FAILED	
Test 08	PASSED	The product warned about a potentially harmful application in its System Modification Attempt dialog. When blocked, the test was unable to install its hook.
Test 09	FAILED	
Test 10	FAILED	
Test 11	PASSED	The product warned about a potentially harmful application in its System Modification Attempt dialog. When blocked, the test was unable to install its hook.
Test 12	PASSED	The product warned about a potentially harmful application in its System Modification Attempt dialog. When blocked, the test was unable to install its hook.
Test 13	FAILED	

Test 14	FAILED	
Test 15	FAILED	

G Data Internet Security 2013 23.0.0.19

Total Score: 0/15

Test Case	Result	Comment
Test 01	FAILED	
Test 02	FAILED	
Test 03	FAILED	
Test 04	FAILED	
Test 05	FAILED	
Test 06	FAILED	
Test 07	FAILED	
Test 08	FAILED	
Test 09	FAILED	
Test 10	FAILED	
Test 11	FAILED	
Test 12	FAILED	
Test 13	FAILED	
Test 14	FAILED	
Test 15	FAILED	

Kaspersky Internet Security 2013 13.0.0.3370

Total Score: 15/15

Test Case	Result	Comment
Test 01	PASSED	Protected browser instances were not infected.
Test 02	PASSED	Protected browser instances were not infected.
Test 03	PASSED	Protected browser instances were not infected.
Test 04	PASSED	Protected browser instances were not infected.
Test 05	PASSED	Protected browser instances were not infected.
Test 06	PASSED	Protected browser instances were not infected.

Test 07	PASSED	Protected browser instances were not infected.
Test 08	PASSED	Credentials were not stolen from the protected browser instance.
Test 09	PASSED	Credentials were not stolen from the protected browser instance.
Test 10	PASSED	Credentials were not stolen from the protected browser instance.
Test 11	PASSED	Credentials were not stolen from the protected browser instance.
Test 12	PASSED	Credentials were not stolen from the protected browser instance.
Test 13	PASSED	Credentials were not stolen from the protected browser instance.
Test 14	PASSED	Credentials were not stolen from the protected browser instance.
Test 15	PASSED	Credentials were not stolen from the protected browser instance.

McAfee Total Protection 2012 11.0.669

Total Score: 0/15

Test Case	Result	Comment
Test 01	FAILED	The machine was unstable after the infection probably due to some kind of protection of McAfee that was not compatible with the technique of this test. The browser crashed very often, sometimes even McAfee processes crashed.
Test 02	FAILED	
Test 03	FAILED	
Test 04	FAILED	
Test 05	FAILED	
Test 06	FAILED	
Test 07	FAILED	
Test 08	FAILED	
Test 09	FAILED	
Test 10	FAILED	
Test 11	FAILED	
Test 12	FAILED	
Test 13	FAILED	
Test 14	FAILED	
Test 15	FAILED	

Microsoft Security Essentials 4.0.1526.0

Total Score: 0/15

Test Case	Result	Comment
Test 01	FAILED	
Test 02	FAILED	
Test 03	FAILED	
Test 04	FAILED	
Test 05	FAILED	
Test 06	FAILED	
Test 07	FAILED	
Test 08	FAILED	
Test 09	FAILED	
Test 10	FAILED	
Test 11	FAILED	
Test 12	FAILED	
Test 13	FAILED	
Test 14	FAILED	
Test 15	FAILED	

Panda Internet Security 2012 17.01.00

Total Score: 1/15

Test Case	Result	Comment
Test 01	FAILED	Sometimes this test caused BSOD.
Test 02	FAILED	
Test 03	FAILED	
Test 04	FAILED	
Test 05	FAILED	
Test 06	FAILED	
Test 07	PASSED	The malicious DLL was not loaded into the browser and so the test's technique was prevented effectively. However, this was probably caused accidentally by Panda's DLLs inside Internet

		Explorer, not by any kind of protection Panda implemented.
Test 08	FAILED	
Test 09	FAILED	
Test 10	FAILED	
Test 11	FAILED	
Test 12	FAILED	
Test 13	FAILED	
Test 14	FAILED	
Test 15	FAILED	

Symantec Norton Internet Security 2012 19.7.1.5

Total Score: 0/15

Test Case	Result	Comment
Test 01	FAILED	
Test 02	FAILED	
Test 03	FAILED	
Test 04	FAILED	
Test 05	FAILED	
Test 06	FAILED	
Test 07	FAILED	
Test 08	FAILED	
Test 09	FAILED	
Test 10	FAILED	
Test 11	FAILED	
Test 12	FAILED	
Test 13	FAILED	
Test 14	FAILED	
Test 15	FAILED	

Trend Micro Titanium Maximum Security 2012 5.2.1035

Total Score: 0/15

Test Case	Result	Comment
Test 01	FAILED	
Test 02	FAILED	
Test 03	FAILED	
Test 04	FAILED	
Test 05	FAILED	
Test 06	FAILED	
Test 07	FAILED	
Test 08	FAILED	
Test 09	FAILED	
Test 10	FAILED	
Test 11	FAILED	
Test 12	FAILED	
Test 13	FAILED	
Test 14	FAILED	
Test 15	FAILED	

Trusteer Rapport 3.5.1201.76

Total Score: 15/15

Test Case	Result	Comment
Test 01	PASSED	Credentials were not stolen. EncryptMessage hook was prevented.
Test 02	PASSED	Credentials were not stolen. EncryptMessage hook was prevented.
Test 03	PASSED	Credentials were not stolen. EncryptMessage hook was prevented.
Test 04	PASSED	Credentials were not stolen. EncryptMessage hook was prevented.
Test 05	PASSED	Credentials were not stolen. EncryptMessage hook was prevented.
Test 06	PASSED	Credentials were not stolen. EncryptMessage hook was prevented.
Test 07	PASSED	Credentials were not stolen. EncryptMessage hook was prevented.
Test 08	PASSED	Credentials were not stolen. Input keys were replaced

		with random characters.
Test 09	PASSED	Credentials were not stolen. Input keys were replaced with random characters.
Test 10	PASSED	Credentials were not stolen. Input keys were replaced with random characters.
Test 11	PASSED	Credentials were not stolen. Input keys were replaced with random characters.
Test 12	PASSED	Credentials were not stolen. Input keys were replaced with random characters.
Test 13	PASSED	Credentials were not stolen. Input keys were replaced with random characters.
Test 14	PASSED	Credentials were not stolen. Input keys were replaced with random characters.
Test 15	PASSED	Credentials were not stolen. Input keys were replaced with random characters.