Panda Adaptive Defense Protection Agent Security Target v3.0

Javier Tallón (https://www.jtsec.es)

2018-02-23

Created by

Table of contents

1 ST Introduction 5
1.1 ST Reference 5
1.2 TOE Reference 5
1.3 TOE Overview 5
1.3.1 Introduction 5
1.3.1.1 Why past methodologies are no longer valid 5
1.3.1.2 A brand new solution 6
1.3.2 TOE Type 7
1.3.3 TOE usage & Major Security Features 7
1.3.4 Non-TOE Hardware/Software/Firmware 8
1.4 TOE Description 9
1.4.1 TOE Logical Scope 9
1.4.1.1 Non-TOE Security Features 10
1.4.2 TOE Physical Scope 11
2 Conformance Claims 12
3 Security Problem Definition 13
3.1 Assets 13
3.2 Threat Agents 13
3.3 Threats to Security 13
3.4 Organizational Security Policies 14
3.5 Assumptions 14
4 Security Objectives 16
4.1 Security objectives for the TOE 16
4.2 Security objectives for the operational environment 17
4.3 Security Objectives Rationale 17
4.3.1 Threats 21
4.3.2 Organizational Security Policies 22
4.3.3 Assumptions 23
5 Security Requirements 25
5.1 Security Functional Requirements 25
5.1.1 FAU: Security audit 25
5.1.1.1 FAU_ARP.1: Security alarms 25
5.1.1.2 FAU_GEN.1/INTERNAL: Audit data generation 25
5.1.1.3 FAU_GEN.1/NDK: Audit data generation 26

5.1.1.4 FAU_SAA.1/VIRUS: Potential violation analysis 27
5.1.1.5 FAU_SAA.3/RISK: Simple attack heuristics 27
5.1.1.6 FAU_SAR.1/NDK: Audit review 27
5.1.1.7 FAU_SAR.3/NDK: Selectable audit review 27
5.1.2 FDP: User data protection 28
5.1.2.1 FDP_SDI.1: Stored data integrity monitoring 28
5.1.3 FIA: Identification and authentication 28
5.1.3.1 FIA_UAU.2: User authentication before any action 28
5.1.3.2 FIA_UID.2: User identification before any action 28
5.1.4 FMT: Security management 28
5.1.4.1 FMT_MOF.1: Management of security functions behaviour 28
5.1.4.2 FMT_SMF.1: Specification of Management Functions 28
5.1.4.3 FMT_SMR.1: Security roles 29
5.1.5 FPT: Protection of the TSF 29
5.1.5.1 FPT_ITC.1: Inter-TSF confidentiality during transmission 29
5.1.5.2 FPT_ITI.1: Inter-TSF detection of modification 29
5.1.5.3 FPT_ITT.1: Basic internal TSF data transfer protection 29
5.1.5.4 FPT_TST.1: TSF testing 29
5.2 Security Assurance Requirements 30
5.3 Security Requirements Rationale 31
5.3.1 Necessity and sufficiency analysis 31
5.3.2 Security Requirement Sufficiency 34
5.3.3 SFR Dependency Rationale 35
5.3.3.1 Table of SFR dependencies 35
5.3.3.2 Justification for missing dependencies 35
5.3.4 SAR Rationale 36
5.3.5 SAR Dependency Rationale 36
5.3.5.1 Table of SAR dependencies 36
6 TOE Summary Specification 37
6.1 Class FAU: Security Audit 37
6.2 Class FDP: User Data Protection 60
6.3 Class FIA: Identification and Authentication 60
6.4 Class FMT: Security Management 60
6.5 Class FPT: Protection of the TSF 60
7 Acronyms 62

8 Glossary of Terms..... 63
9 Document References..... 64

1 ST Introduction

1.1 ST Reference

Title: Panda Adaptive Defense Protection Agent Security Target

Version: v3.0

Author: Javier Tallón (https://www.jtsec.es)

Date of publication: 2018-02-23

1.2 TOE Reference

TOE Name: Panda Adaptive Defense Protection Agent

TOE Version: 8.0

TOE Developer: Panda Security

1.3 TOE Overview

1.3.1 Introduction

1.3.1.1 Why past methodologies are no longer valid

We are lately seeing blogs attempting to publicly demonstrate that next-generation protection solutions, like Adaptive Defense, are vulnerable. These proofs of concept aim to demonstrate that there are malicious files that evade detection when reaching a system or attempting to run. The problem with these demonstrations is that the writer expects the malicious files to be stopped before being run. But that's a mistake, and reveals a clear misunderstanding of this new protection model based on the continuous monitoring of process activities.

To be truly effective, a next-generation solution must provide continuous protection against all types of attacks. This means that it must offer continuous prevention, detection at runtime, visibility into every action taken, and intelligence to block malicious actions such as lateral movements. It is not enough to provide detection at file level based on a list of malware files. Efficient security means being able to protect systems before, after and during an attack.

The cyber-security 'war' goes beyond the 'battle' of detecting malicious files when they reach a computer or attempt to run. It will be won by whoever is capable of efficiently, seamlessly and unobtrusively monitoring every process running on devices, blocking those that, despite being apparently and initially harmless, show malicious behaviors. Today's malware is extremely sophisticated and should never be underestimated.

But not only that. Protection is not only about detecting threats before, after and during an attack, it is also remediation and prevention.

That's why a next-generation solution must also include response and remediation capabilities. These products are known in the security sector as EDR (Endpoint Detection and Response) solutions, and they incorporate forensic analysis tools capable of tracing every action taken on the endpoint in order to remediate and prevent present and future attacks.

Panda Adaptive Defense integrates all of those features into a single Next-Generation protection solution based on continuous monitoring, and which provides prevention, detection, visibility and intelligence to block known and unknown attacks. In addition to continuous monitoring via hundreds of sensors, Adaptive Defense also provides forensic analysis tools for efficient remediation and prevention.

When you read these proofs of concept, you must understand that they are not real. The fact that a security solution doesn't detect a file as malware at the time of reaching a system doesn't mean that it is not efficient. On the contrary, in the particular case of Adaptive Defense, it is perfectly possible that the solution doesn't detect the file at that time, but it will detect it as soon as it attempts to run, or will monitor and block it during an attack.

This ability is not present in traditional solutions based on a more or less generalist malware blacklisting strategy, and which rely on detecting malicious files on the system or when attempting to run. With these solutions, if a malicious file is not classified as malware, it will be allowed to run regardless of the actions it carries out during its life cycle.

Adaptive Defense might also let it run, albeit keeping an eye on it at all times and reporting its activities to our Machine Learning Intelligence platform. This system, which is in constant evolution and correlates data from millions of endpoints with hundreds of sensors, will determine if the file's activities constitute malicious behavior, in which case it will prevent it from running. Then, the file will be immediately classified either automatically or by a team of cyber-security experts. This analysis will determine with complete accuracy the nature of the attack. The old model doesn't provide any of this.

Welcome to the Next-Generation Panda Security!

1.3.1.2 A brand new solution

The Adaptive Defense security system provides a protection mechanism to the device in which it is installed based on the management of the applications running on that device.

Its objective is to maintain the maximum level of protection against external and internal threats, based, on the one hand, on the continuous monitoring of the different processes that are executed in the device and, on the other, in the automatic classification of any executable code you see on your computer, carried out through a correlation system.

The system consists of a protection agent installed in the device (the TOE), an expert system of external classification running in Panda Security's cloud environment, a communications agent that performs the managements and remote maintenance tasks, and an external management system that allows an administrator to manage the protection agent through a web console.

By itself, the continuous monitoring capability offered by the protection agent allows the implementation of a forensic audit model, which helps the network administrator to detect the origin and method employed by the potential threats and therefore allows the creation and implementation of Security policies that reinforce the strength of IT infrastructure.

Besides, thanks to the automated classification of applications, security models based on risk management can be implemented. The expert system models the executable code defining for each element a level of risk and a confidence in each diagnosis. The classification of each application is carried out jointly by means of local technologies as well as the analysis of external correlation. Ultimately, every element is classified as trusted or untrusted and therefore its execution is allowed or not.

The system implements several tolerance levels to cover the different protection scenarios based on the actions performed on the elements for which the system is calculating the level of risk. Based on these different levels of risk tolerance, the system can implement from negative models, where it only blocks what is known a priori that is malicious, to positive models, where it blocks everything until it is classified as trusted.

1.3.2 TOE Type

The TOE is a Next Generation Endpoint Protection software with risk analysis and cloud capabilities.

1.3.3 TOE usage & Major Security Features

The TOE is installed in a desktop workstation or server and automatically provides the following features:

- Interception of the operating system picking up all operations performed by the applications.
- Sending a record of the operations performed by each process to the correlation system, saving its response about the classification of the application and its modules
- Use of the correlation system response, determining the action to be taken on each application
- Use of its local capabilities based on rules of behavior and known malware signatures to determine the action to be taken on each application
- Execution of the derivated action on the monitored application: block or allow its execution or load based on the two before.
- Detect and forbid exploitation techniques
- Detect and forbid access to malicious websites
- Detect and allow/deny the reading/writing to removable devices
- Generation of notifications on the action taken.

- Self-protection from malicious process

1.3.4 Non-TOE Hardware/Software/Firmware

The TOE needs for its execution the following software that is bundled in the same product or provided through a network connection:

- Local console: This is the local user interface of the Protection Agent (TOE)
- Management Agent: It is a service that manages the Protection Agent (TOE) through communications with the cloud management services, it also download and install the Protection Agent (TOE).
- Panda Remote Management (Cloud Management Services): Web services hosted in the cloud whose purpose is the administration of the protections in a simple and centralized way. They work in a passive way, where the management agent request new information to determine if new administrative tasks need to be undertaken.
- Panda Collective Intelligence: Web services hosted in the private cloud of Panda that give access to the Collective Intelligence of Panda. Its main functions are the collection, classification, and correlation of samples of the whole community.

The TOE runs in a General Purpose Computer using a Microsoft Windows 7 Operating system or above.

Specifically, it requires as a minimum:

- Processor: Pentium 300 MHz or equivalent
- RAM: 256MB
- Space for installation: 650MB
- Browser: Internet Explorer 6.0 or later

If installed in a Workstation:

  - Operating Systems: Windows 10, Windows 8.1, Windows 8, Windows 7 (32-bit and 64-bit).
  - RAM: For the Antivirus protection: 64MB

If installed in a Server:

  - Operating Systems: Windows Server 2008(32-bit and 64-bit) , Windows Server 2008 R2 (32-bit and 64-bit), Windows Server 2012 and Windows Server 2012 R2
  - RAM: 256MB

The TOE can also run on a virtualized platform:

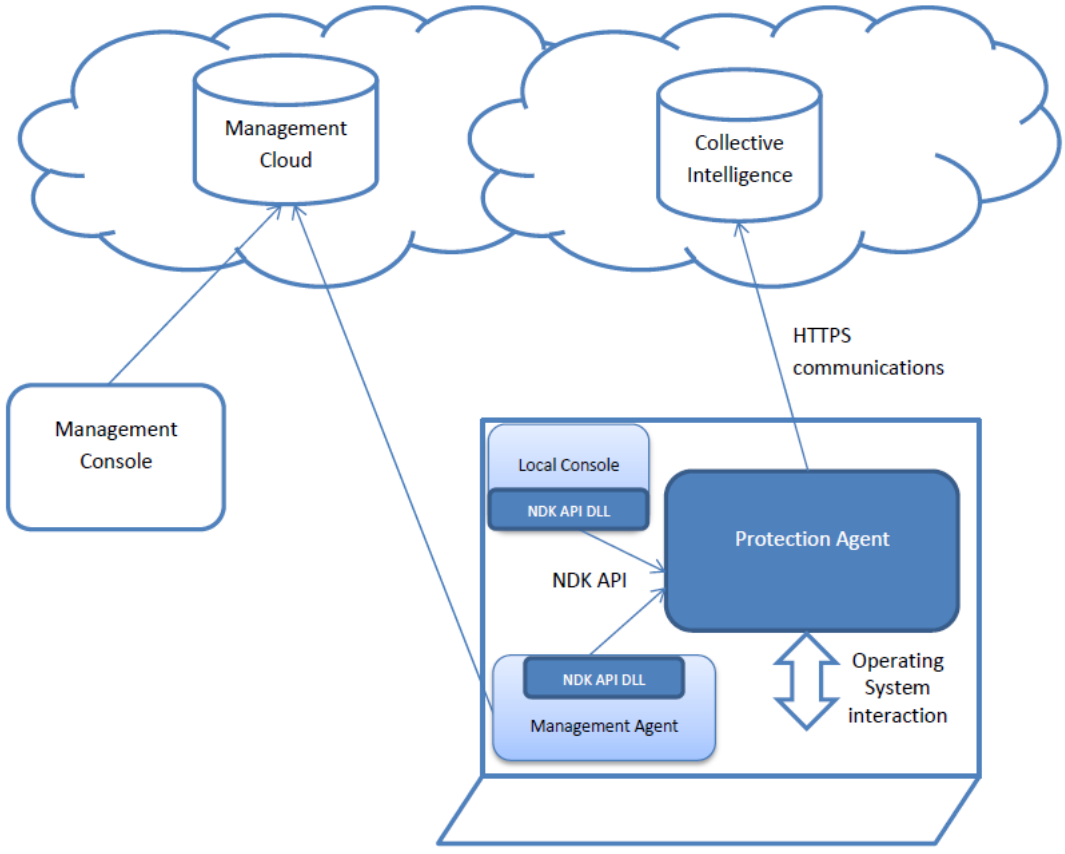
  - VMWare ESX 3.x, 4.x, 5.x and 6.x

- VMWare Workstation 6.0, 6.5, 7.x, 8.x, 9.x, 10.x, 11.x and 12.x
- Virtual PC 6.x
- Microsoft Hyper-V Server 2008, 2008R2, 2012, 2012R2 and 2016 3.0
- Citrix XenDesktop 5.x, XenClient 4.x, XenServer and XenApp 5.x and 6.x

To provide its security functionality, the TOE uses the WinHTTP v5.1 service provided by all the above-mentioned operating systems.

1.4 TOE Description

1.4.1 TOE Logical Scope



The TOE, marked as dark blue, is composed of the Protection Agent, which implements the main TOE security functionality (process interception, risk analysis, audit functionality, etc), and the NDKAPI DLLs, which provide an ease-of-use API, using an encrypted channel, to other Panda processes in order to receive notifications and user logs from the Protection Agent and to manage the Protection Agent itself.

The NDKAPI DLL authenticates the loading process so that only allowed processes like the Management Agent (which receives information from the cloud management console) and the Local Console can interact with the Protection Agent.

The Protection Agent place several user and kernel hooks in the operating system to allow monitoring of every launched process and will allow or deny the execution of a process or functionality based on:

- Information obtained from the Collective Intelligence through an HTTPS connection, which has previously received the monitored processes actions.
- Malware signatures
- Exploit signatures
- Access to malicious website
- Process behavior local risk analysis
- Whitelisted websites (Panda maitains a list of phishing or malware urls so they are blocked unless whitelisted)
- Device control configuration
- Configured risk analysis operation mode

1.4.1.1 Non-TOE Security Features

As any AV, the TOE relies on the intercepted operating system. As expected, cryptographic and network functionality is provided by the OS, interfaces that the TOE use in a secure manner.

The TOE uses for its correct operation the collective intelligence stored in the private cloud of Panda Security. The calculations made by the Panda Security private cloud are out of the scope of this evaluation, although it is subject to evaluation the security of this communications channel.

The TOE also provides the following features that are not subject to evaluation in this Security Target:

- Firewall: protection based on Allow/Deny rules and IDS capabilities
- Web URL filtering: Categorization and filtering navigation based on URL ranking
- Microsoft Exchange Server Protection: Anti malware and anti spam protection for Exchange servers

1.4.2 TOE Physical Scope

The TOE is distributed to the final client as part of a bigger product where usually the management agent will download and install the Local Console and the Protection Agent + NDK (TOE) from Panda Servers.

The TOE consists of the following executables and guidance documents:

Type	Name	Description
Software	PSANHost.exe	Principal component
Software	NdkApi.dll	Implements communications security to manage the TOE
Software	PSINFile.sys, PSINProc.sys, PSINReg.sys, PSINKNC.sys, PSINAflt.sys, PSINProt.sys, dvctprov.sys, PSINDvct.sys, NNSNAHSL.sys, NNSpihsw.sys, NNSStrm.sys, NSSmtp.sys, NNSPop3.sys, NNSHttp.sys, NNSHttps.sys, NSDHCP.sys, NNSAlpc.sys, NNSIds.sys, NNSpicc.sys, NNSProt.sys, NNSPrv.sys, NNStlsc.sys	Drivers to monitor system-wide actions and behaviors
Document	Operational Guidance v2.0	Manuals for secure TOE operation
Document	Preparative Guidance v3.0	Manuals for secure TOE installation
Document	NDK API v2.2.0.2	NDK API use guideance
Document	Functional Specification v2.0	TOE Functional Specification
Document	Adaptive Defense 360 Guide for network administrators v2.3.5	Manual for product operation

2 Conformance Claims

This Security Target and the TOE described are in accordance with the requirements of Common Criteria 3.1R4.11

This Security Target claims conformance with the following parts of Common Criteria:

- Conformance with [CC31R4P2].
- Conformance with [CC31R4P3].

The methodology to be used for the evaluation is described in the "Common Evaluation Methodology" of the Common Criteria standard of September 2012, version 3.1 revision 4 with an evaluation assurance level of EAL2 + ALC_FLR.1.

This Security Target does not claim conformance with any protection profile.

3 Security Problem Definition

This section describes the security aspects of the operational environment and its expected use in said environment. It includes the declaration of the TOE operational environment that identifies and describes:

- The alleged known threats that will be countered by the TOE
- The organizational security policies that the TOE has to adhere to
- The TOE usage assumptions in the suggested operational environment.

We will begin defining Assets and Agents of threats.

3.1 Assets

A.TOE_FUNCTIONALITY: The TOE normal behaviour and security functionality is an asset itself because its self-protection capabilities.

A.WORKSTATION: The normal operation of the workstation where the TOE is installed and the absence of virus and other forms of malware.

3.2 Threat Agents

TA.ATTACKER: A malicious attacker with basic attack potential or process acting on behalf an attacker with basic attack potential.

TA.MALWARE: A known piece of malware or virus which has been publicly analyzed and classified and whose signature is well known.

TA.UNKNOWN: An unknown piece of malware that is still not well known but whose behavior can be classified as offensive under a certain risk level.

TA.EXPLOIT: A known exploitation technique that has been modeled in the anti-exploit capabilities of the TOE

3.3 Threats to Security

This section identifies the threats to assets that require protection by the TOE. The threats are defined in terms of assets concerned, attackers and the adverse action that materializes the threat.

T.DISABLE: **TA.ATTACKER** gets to disable some of the TOE functionality **A.TOE_FUNCTIONALITY** through an unintended use of the NDKAPI or modifying the TOE itself or some of its configuration files.

T.VIRUS: A known **TA.MALWARE** infects **A.WORKSTATION** bypassing TOE security functionality.

T.UNKNOWN: An unknown malware or payload **TA.UNKNOWN** is able to execute an exploit or malicious code in **A.WORKSTATION** even when the risk level that characterizes the process according to its execution log and the rules that the TOE applies, is lower that the risk level that allows code execution according to the risk tolerance level configured in the TOE.

T.EXPLOIT: An exploit **TA.EXPLOIT** is able to execute its payload in the context of **A.WORKSTATION**

3.4 Organizational Security Policies

The organizational Security policies are defined as follows.

P.AUDIT: Audit data will be made available through the NDK API for their consultation on possible important events in the operation.

P.NOTIFY: The client applications of the TOE may be able to subscribe for important events using the NDK API of the TOE.

P.IC_FEED: Execution traces of every process execution will be sent through Internet to the Panda Security Cloud to correlate events and provide Collective Intelligence.

P.MANAGEMENT: The TOE shall allow its own management through the NDK API in order to tailor security parameters like the risk operation mode.

P.DEVICE: The TOE will allow/deny the read/write access to removable devices classes

P.WEB: The TOE will deny access to not whitelisted websites that are ranked as containing malware or phishing.

3.5 Assumptions

The assumptions when using the TOE are the following:

A.INSTALL: The installation will be performed by qualified personnel on an uninfected equipment and whose configuration has not been altered to pervert the functionality of the TOE.

A.NOEVIL: Sites using the TOE shall ensure that authorized administrators are nonhostile, appropriately trained and follow all administrator guidance.

A.PHYSICAL: Physical security will be provided within the domain for the value of the IT assets protected by the TOE and the value of the stored, processed, and transmitted information.

A.INTERNET: The environment will provide a regular connection with the Panda Security cloud servers for the correct update of the product, the databases of signatures of the same, and access to the servers of Collective Intelligence.

A.OS: The OS and hardware will faithfully execute the commands of the TOE, and will not tamper with the TOE in any manner, interfaces through which the TOE obtains primitive functionality from the OS and hardware (executing machine code instructions, OS APIs, time and date, ...) will not be accessible, since the OS/hardware are the only entities that can access that interface, and they are completely trusted.

A.RESPONSIBLE_NDK: The NDK interface will be operated only by Panda workforce or other close associate companies who will be trusted and adequately trained, and will not try to abuse the API.

4 Security Objectives

The security objectives are high level declarations, concise and abstract of the solution to the problem exposed in the former section, which counteracts the threats and fulfills the security policies and the assumptions. These consist of:

- the security objectives for the operational environment.
- the security objectives for the TOE

4.1 Security objectives for the TOE

The security objectives for the TOE must determine (to the desired extent) the responsibility of the TOE in countering the threats and in enforcing the OSPs. Each objective must be traced back to aspects of identified threats to be countered by the TOE and to aspects of OSPs to be met by the TOE.

O.VIRUS: The TOE will detect and take action against known viruses introduced to the workstation.

O.SELF_PROTECTION: The TSF will protect itself and its resources from external interference, tampering. If configured to do so, it will prevent product uninstall.

O.INTERCEPTION: The TOE will intercept the execution of every process in order to monitor its behavior and deny dangerous operations

O.NOTIFY: The TOE will notify the detected attack or other important events to subscribed entities.

O.UPDATE: The TOE will verify the integrity of the signatures when updating.

O.RISK: The TOE will register the actions executed by a process allowing or denying its execution based on its behavior during its entire lifecycle and Panda collective intelligence and TOE mode of operation.

O.AUDIT: The TOE will audit important events and allow the TOE NDK API clients to review them.

O.IC_FEED: The TOE will send to the collective intelligence the actions performed by the monitored process in a secure way.

O.AUTHENTICATION: The TOE will authenticate the clients of the NDK API

O.NONDISCLOSE: The TOE will prevent disclosure of communications of the NDK API

O.MANAGEMENT: The TOE allows its own management through the NDK API in order to tailor security parameters like the risk threshold.

O.EXPLOIT: The TOE will implement anti exploit features for known exploits techniques

4.2 Security objectives for the operational environment

The security objectives for the Operational Environment determine the responsibility of the environment in countering the threats, enforcing the OSPs and upholding the assumptions. Each objective must be traced back to aspects of identified threats to be countered by the environment, to aspects of OSPs to be enforced by the environment and to assumptions to be uphold by the environment.

OE.INSTALL: The installation will be performed by qualified personnel on an uninfected equipment and whose configuration has not been altered to pervert the functionality of the TOE.

OE.NOEVIL: Sites using the TOE shall ensure that authorized administrators are nonhostile, appropriately trained and follow all administrator guidance.

OE.PHYSICAL: Physical security will be provided within the domain for the value of the IT assets protected by the TOE and the value of the stored, processed, and transmitted information.

OE.INTERNET: The environment will provide a regular connection with the Panda Security cloud servers for the correct update of the product, the databases of signatures of the same, and access to the servers of Collective Intelligence.

OE.OS: The OS and hardware will faithfully execute the commands of the TOE, and will not tamper with the TOE in any manner, interfaces through which the TOE obtains primitive functionality from the OS and hardware (executing machine code instructions, OS APIs, cryptographic functionality, time and date, ...) will not be accessible, since the OS/hardware are the only entities that can access that interface, and they are completely trusted.

OE.RESPONSIBLE_NDK: The NDK interface will be operated only by Panda workforce or other close associate companies who will be trusted and adequately trained, and will not try to abuse the API.

4.3 Security Objectives Rationale

The following table provides a mapping of security objectives tracing each security objective for the TOE back to threats countered by that security objective and OSPs enforced by that security objective, and each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective. This illustrates that the security objectives counter all threats, the security objectives enforce all OSPs and the security objectives for the operational environment uphold all assumptions.

	O.VIRUS	O.SELF_PROTECTION	O.INTERCEPTION	O.NOTIFY	O.UPDATE	O.RISK	O.AUDIT	O.IC_FEED	O.AUTHENTICATION	O.NONDISCLOSE	O.MANAGEMENT	O.EXPLOIT	OE.INSTALL	OE.NOEVIL	OE.PHYSICAL	OE.INTERNET	OE.OS	OE.RESPONSIBLE_NDK
T.DISABLE		X	X						X	X			X	X	X	X	X	X
T.VIRUS	X		X		X											X		
T.UNKNOWN			X			X		X								X		
T.EXPLOIT			X					X				X				X		
P.AUDIT							X										X	
P.NOTIFY				X														
P.IC_FEED			X					X								X	X	
P.MANAGEMENT											X							
P.DEVICE			X															
P.WEB			X		X											X		

	O.VIRUS	O.SELF_PROTECTION	O.INTERCEPTION	O.NOTIFY	O.UPDATE	O.RISK	O.AUDIT	O.IC_FEED	O.AUTHENTICATION	O.NONDISCLOSE	O.MANAGEMENT	O.EXPLOIT	OE.INSTALL	OE.NOEVIL	OE.PHYSICAL	OE.INTERNET	OE.OS	OE.RESPONSIBLE_NDK
A.INSTALL													X					
A.NOEVIL														X				
A.PHYSICAL															X			
A.INTERNET																X		
A.OS																	X	
A.RESPONSIBLE_NDK																		X

Table 1 *Security Objectives vs Security Problem Definition*

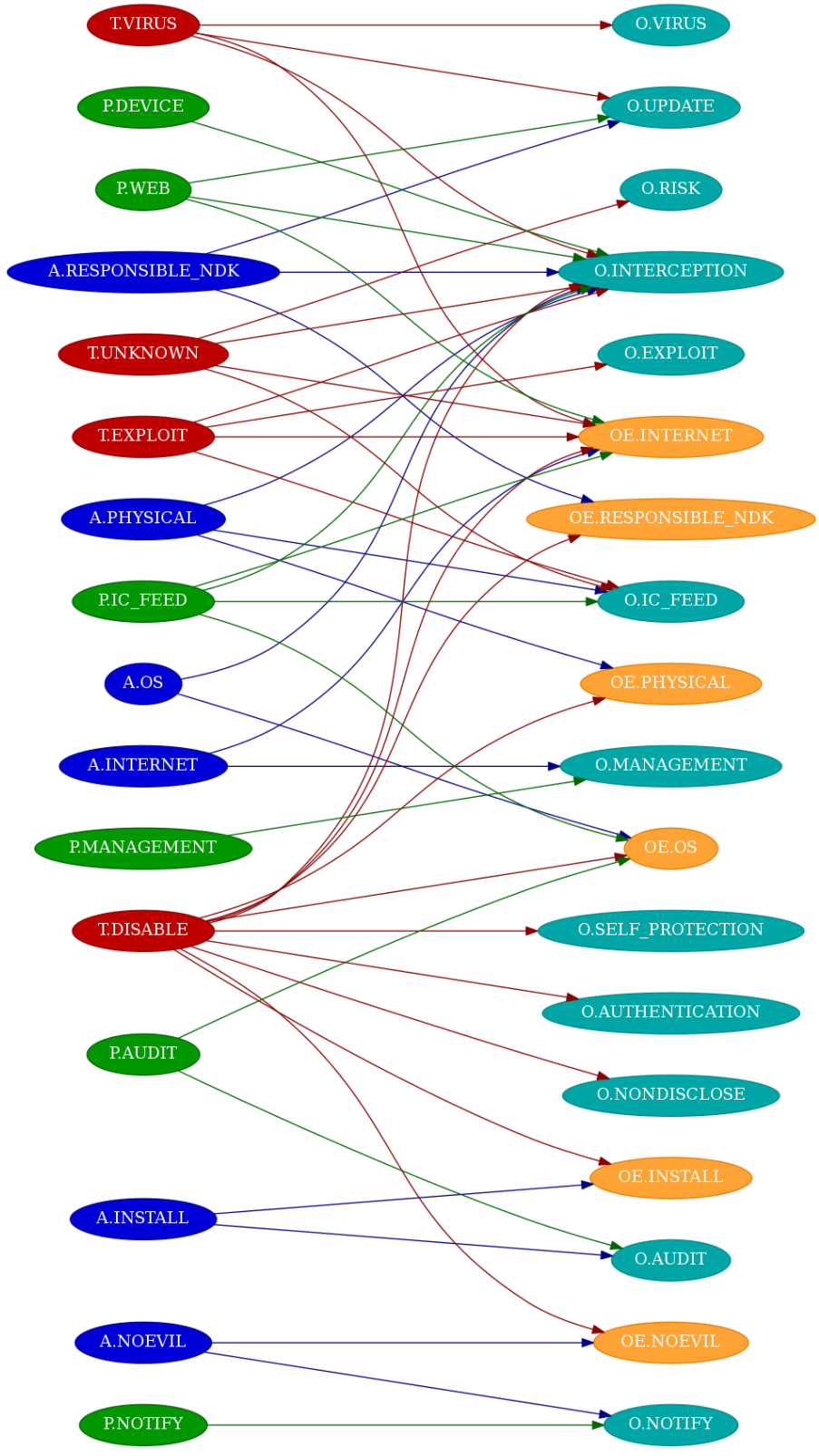


Figure 1 *Mapping of Security Problem Definition to Security Objectives*

4.3.1 Threats

T.DISABLE: The TOE combined security objectives **O.SELF_PROTECTION** and **O.INTERCEPTION** mitigate this threat preventing the TOE and its configuration files modification.

O.NONDISCLOSE and **O.AUTHENTICATION** prevent NDK API channels disclosure and connection from unauthenticated parties respectively.

OE.RESPONSIBLE_NDK prevents misbehavior from users of the API.

The security objectives for the operational environment **OE.NOEVIL** contributes to mitigate this threat preventing administrators to disable the TOE. **OE.PHYSICAL** contributes disallowing physical access to attackers. **OE.INSTALL** ensures that no malicious code has been installed previous to the TOE installation, while **OE.INTERNET** provides updated signatures and behavioral rules. **OE.OS** contributes so the Operating System behaves as expected.

T.VIRUS: **O.VIRUS** and **O.INTERCEPTION** directly mitigate this threat blocking execution of perverse known processes with the help of **O.UPDATE** and **OE.INTERNET** which ensure the availability of updated signatures.

T.UNKNOWN: The TOE security objective **O.INTERCEPTION** mitigate this threat preventing the execution of a process whose behavior is risky according to the TOE configured threshold and the information obtained through **O.RISK** .

The data obtained through **OE.INTERNET** helps mitigating this threat by providing updated signatures and rules and information from Panda Security Collective Intelligence which also receives the running processes behavior through **O.IC_FEED**.

T.EXPLOIT: The TOE security objective **O.INTERCEPTION** mitigate this threat preventing the execution of a process that is being exploiting according to the signatures managed by **O.EXPLOIT**.

The data obtained through **OE.INTERNET** helps mitigating this threat by providing updated signatures and rules and information from Panda Security Collective Intelligence which also receives the running processes behavior through **O.IC_FEED**.

The following table maps the threats of the security problem established to the security objectives of the TOE and the security objectives of the operational environment.

Threats	Security Objectives
T.DISABLE	O.SELF_PROTECTION O.INTERCEPTION O.NONDISCLOSE O.AUTHENTICATION OE.RESPONSIBLE_NDK OE.NOEVIL OE.PHYSICAL

Threats	Security Objectives
	OE.INSTALL OE.INTERNET OE.OS
T.VIRUS	O.VIRUS O.INTERCEPTION O.UPDATE OE.INTERNET
T.UNKNOWN	O.INTERCEPTION O.RISK O.IC_FEED OE.INTERNET
T.EXPLOIT	O.INTERCEPTION O.EXPLOIT O.IC_FEED OE.INTERNET

Table 2 Threats vs Security Objectives

4.3.2 Organizational Security Policies

P.AUDIT: The security objective for the TOE**O.AUDIT** directly implements this organisational security policy with the help of **OE.OS** which provides logging storage capabilities.

P.NOTIFY: The security objective for the TOE**O.NOTIFY** directly implements this organisational security policy by sending notifications to the subscribed processes using NDKAPI.

P.IC_FEED: The security objective **O.IC_FEED** directly implement this policy with the help of the process logging capabilities described in **O.INTERCEPTION** by sending these logs to the Panda Security cloud servers in order to provide collective intelligence to other instances of the TOE. **OE.OS** contributes behaving in the expected way and **OE.INTERNET** providing connection to the cloud.

P.MANAGEMENT: The security objective **O.MANAGEMENT** directly implements this policy allowing TOE management through the NDKAPI.

P.DEVICE: **O.INTERCEPTION** implements this policy controlling access to devices through interception of system calls.

P.WEB: **O.INTERCEPTION** implements this policy controlling access to known malicious websites with the help of **O.UPDATE** and **OE.INTERNET** which ensure the availability of updated signatures.

The following table maps the organisational security policies of the problem established to the security objectives of the TOE and the security objectives of the operational environment.

OSPs	Security Objectives
P.AUDIT	O.AUDIT OE.OS
P.NOTIFY	O.NOTIFY
P.IC_FEED	O.IC_FEED O.INTERCEPTION OE.OS OE.INTERNET
P.MANAGEMENT	O.MANAGEMENT
P.DEVICE	O.INTERCEPTION
P.WEB	O.INTERCEPTION O.UPDATE OE.INTERNET

Table 3 *OSPs vs Security Objectives*

4.3.3 Assumptions

A.INSTALL: The security objective for the operational environment **OE.INSTALL** implements directly this assumption.

A.NOEVIL: The security objective for the operational environment **OE.NOEVIL** implements directly this assumption.

A.PHYSICAL: The security objective for the operational environment **OE.PHYSICAL** implements directly this assumption.

A.INTERNET: The security objective for the operational environment **OE.INTERNET** implements directly this assumption.

A.OS: The security objective for the operational environment **OE.OS** implements directly this assumption.

A.RESPONSIBLE_NDK: The security objective for the operational environment **OE.RESPONSIBLE_NDK** implements directly this assumption.

The following table maps the assumptions of the problem established to the security objectives of the TOE and the security objectives of the operational environment.

Assumptions	Security Objectives
A.INSTALL	OE.INSTALL
A.NOEVIL	OE.NOEVIL
A.PHYSICAL	OE.PHYSICAL
A.INTERNET	OE.INTERNET
A.OS	OE.OS
A.RESPONSIBLE_NDK	OE.RESPONSIBLE_NDK

Table 4 Assumptions vs Security Objectives for the Operational Environment

5 Security Requirements

This section defines the Security functional requirements (SFRs) and the Security assurance requirements (SARs) that fulfill the TOE. Assignment, selection and iteration operations have been made, adhering to the following conventions:

- Assignments. The word “assignment” is maintained and the resolution is presented in boldface, italic and blue color.
- Selections. The word “selection” is maintained and the resolution is presented in boldface, italic and blue color.
- Iterations. It includes “/” and an “identifier” following requirement identifier that allows to distinguish the iterations of the requirement. Example: FCS_COP.1/XXX.
- Refinements: The text were the refinement has been done is shown bold, italic and blue color.

5.1 Security Functional Requirements

5.1.1 FAU: Security audit

5.1.1.1 FAU_ARP.1: Security alarms

FAU_ARP.1.1 The TSF shall take ***[assignment: block process execution and notify subscribed clients of the NDK API]*** upon detection of a potential security violation.

5.1.1.2 FAU_GEN.1/INTERNAL: Audit data generation

FAU_GEN.1.1 The TSF shall be able to generate an audit record of the following auditable events:

a) Start-up and shutdown of the audit functions;

b) All auditable events for the ***[selection: not specified]*** level of audit; and

c) ***[assignment: - Alert table (detected threats)***
- Install table (information generated during the installation/uninstallation)
- Monitoredopen table (Data files accessed by the applications)
- MonitoredRegistry table (every attempt to modify the registry as well as registry accesses related to permissions, passwords, certificate stores and other)
- Notblocked table (items that Adaptive Defense has not scanned due to exceptional situations such as service timeout on startup, configuration changes, etc.)
- Ops Table (operations performed by the processes)
- ProcessNetBytes Table (network data usage of the processes)
- Registry table (operations performed on the registry branches used by malicious programs to become persistent and survive computer restarts.)
- Socket table (network connections established by the processes)

- Toast table (logs an entry every time the agent shows a message to the customer)
- ToastBlocked table (contains a record for each blocked process, as Adaptive Defense has not yet returned the relevant classification.)
- URLdownload table (information on the HTTP downloads performed by the processes)
- VulnerableAppsFound table (logs every vulnerable application found)] .

FAU_GEN.1.2 The TSF shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity (if applicable), and the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, ***[assignment: none]*** .

Application Note

This SFR shall not be confused with FAU_GEN.1/NDK. The audited events cannot be accessed through the NDK API (they are internal) but rather they are directly sent to the Collective Intelligence or processed to determine the actions to be taken.

5.1.1.3 FAU_GEN.1/NDK: Audit data generation

FAU_GEN.1.1 The TSF shall be able to generate an audit record of the following auditable events:

a) Start-up and shutdown of the audit functions;

b) All auditable events for the ***[selection: not specified]*** level of audit; and

c) ***[assignment: - Analysis status***
- Malware detected
- Grayware detected
- Quarantine error notifications
- Quarantine restoration events
- AV Status Event
- Quarantine reclassifications
- KRE notifications
- Autodiagnosis notification
- Update notification
- Antiexploit actuation result
- Device control event
- Synchronization status
- Shows untrusted file toast
- Shows blocked untrusted file toast
- Ask to user for kill process
- Antiexploit notification
- Meta Exploit Alert
- Shows executed program that is included in black list] .

FAU_GEN.1.2 The TSF shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity (if applicable), and the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, ***[assignment: none]*** .

Application Note

This requirement is intended to describe the most important events registered by the TOE in order to notify the user about the TOE behaviour. It will register those events in Windows' EventLog or directly send a notification to subscribed users without further storage.

5.1.1.4 FAU_SAA.1/VIRUS: Potential violation analysis

FAU_SAA.1.1 The TSF shall be able to apply a set of rules in monitoring the audited events and based upon these rules indicate a potential violation of the enforcement of the SFRs.

FAU_SAA.1.2 The TSF shall enforce the following rules for monitoring audited events:

a) Accumulation or combination of ***[assignment: matching signatures, patterns and heuristics]*** known to indicate a potential security violation;

b) ***[assignment: none]*** .

5.1.1.5 FAU_SAA.3/RISK: Simple attack heuristics

FAU_SAA.3.1 The TSF shall be able to maintain an internal representation of the following signature events ***[assignment: none]*** that may indicate a violation of the enforcement of the SFRs.

Application Note

Only the risk analysis capabilities of the TOE are considered under this SFR, so "none" is chosen in the assignment operation.

FAU_SAA.3.2 The TSF shall be able to compare the signature events against the record of system activity discernible from an examination of ***[assignment: process activity (I/O, network, file system, system calls) and TOE configured operation mode]*** .

FAU_SAA.3.3 The TSF shall be able to indicate a potential violation of the enforcement of the SFRs when a system event is found to match a signature event that indicates a potential violation of the enforcement of the SFRs.

5.1.1.6 FAU_SAR.1/NDK: Audit review

FAU_SAR.1.1 The TSF shall provide ***[assignment: NDK API clients]*** with the capability to read ***[assignment: FAU_GEN.1/NDK logs]*** from the audit records.

FAU_SAR.1.2 The TSF shall provide the audit records in a manner suitable for the user to interpret the information.

Application Note

While the TOE provides audit review capabilities, this information is accessed through the NDK API, and therefore it is not expected to be used by an human user, but by a piece of software, the management agent, which will send the information to the cloud.

5.1.1.7 FAU_SAR.3/NDK: Selectable audit review

FAU_SAR.3.1 The TSF shall provide the ability to apply ***[assignment: selection of beginning date, end date and beginning id]*** of audit data based on ***[assignment: date and id]*** .

5.1.2 FDP: User data protection

5.1.2.1 FDP_SDI.1: Stored data integrity monitoring

FDP_SDI.1.1 The TSF shall monitor user data stored in containers controlled by the TSF for ***[assignment: byte modification]*** on all objects, based on the following attributes: ***[assignment: MD5 value of the update file]*** .

5.1.3 FIA: Identification and authentication

5.1.3.1 FIA_UAU.2: User authentication before any action

FIA_UAU.2.1 The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

5.1.3.2 FIA_UID.2: User identification before any action

FIA_UID.2.1 The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

5.1.4 FMT: Security management

5.1.4.1 FMT_MOF.1: Management of security functions behaviour

FMT_MOF.1.1 The TSF shall restrict the ability to ***[selection: determine the behaviour of, disable, enable, modify the behaviour of]*** the functions ***[assignment: described in FMT_SMF.1]*** to ***[assignment: NDK API users]*** .

5.1.4.2 FMT_SMF.1: Specification of Management Functions

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions: ***[assignment: - Autodiagnostic***
- Audit review
- Quarentine management
- Event subscription
- Enable / Disable AV protection
- Enable / Disable different filters
- On-demand analysis
- File/directory exclusions
- Whitelists & Blacklists
- Enable / Disable web protection

- Access to removable devices
- Enable / Disable Security Risk Protection
- Security Risk Protection working mode
- Enable / Disable Security Risk Protection alerts] .

5.1.4.3 FMT_SMR.1: Security roles

FMT_SMR.1.1 The TSF shall maintain the roles ***[assignment: NDK API User]*** .

FMT_SMR.1.2 The TSF shall be able to associate users with roles.

Application Note

From the TOE perspective, there is only one role applicable to every process accessing the NDK API.

5.1.5 FPT: Protection of the TSF

5.1.5.1 FPT_ITC.1: Inter-TSF confidentiality during transmission

FPT_ITC.1.1 The TSF shall protect all TSF data transmitted from the TSF to another trusted IT product from unauthorised disclosure during transmission.

5.1.5.2 FPT_ITI.1: Inter-TSF detection of modification

FPT_ITI.1.1 The TSF shall provide the capability to detect modification of all TSF data during transmission between the TSF and another trusted IT product within the following metric: ***[assignment: a single message authentication code error during transmission]*** .

FPT_ITI.1.2 The TSF shall provide the capability to verify the integrity of all TSF data transmitted between the TSF and another trusted IT product and perform ***[assignment: discontinuation of the communication channel or disregard of the data]*** if modifications are detected.

5.1.5.3 FPT_ITT.1: Basic internal TSF data transfer protection

FPT_ITT.1.1 The TSF shall protect TSF data from ***[selection: disclosure]*** when it is transmitted between separate parts of the TOE.

Application Note

This requirement address the internal communication between the two separated parts of the TOE, the protection agent itself and the NDK API DLL.

5.1.5.4 FPT_TST.1: TSF testing

FPT_TST.1.1 The TSF shall run a suite of self tests ***[selection: during initial start-up, periodically during normal operation]*** to demonstrate the correct operation of ***[selection: [assignment: - Driver status***
- Protection status

- Internet Connection status
- Connection to IC status]] .

FPT_TST.1.2 The TSF shall provide authorised users with the capability to verify the integrity of ***[selection: [assignment: none]]*** .

FPT_TST.1.3 The TSF shall provide authorised users with the capability to verify the integrity of ***[selection: [assignment: none]]*** .

5.2 Security Assurance Requirements

The development and the evaluation of the TOE shall be done in accordance to the following security assurance requirements: **EAL2 + ALC_FLR.1**

The following table shows the assurance requirements by reference the individual components in [CC31R4P3]

Assurance Class	Assurance Components
ASE: Security Target evaluation	ASE_CCL.1: Conformance claims ASE_ECD.1: Extended components definition ASE_INT.1: ST introduction ASE_TSS.1: TOE summary specification ASE_OBJ.2: Security objectives ASE_REQ.2: Derived security requirements ASE_SPD.1: Security problem definition
ALC: Life-cycle support	ALC_CMC.2: Use of a CM system ALC_CMS.2: Parts of the TOE CM coverage ALC_DEL.1: Delivery procedures ALC_FLR.1: Basic flaw remediation
ADV: Development	ADV_ARC.1: Security architecture description ADV_FSP.2: Security-enforcing functional specification ADV_TDS.1: Basic design
AGD: Guidance documents	AGD_OPE.1: Operational user guidance AGD_PRE.1: Preparative procedures
ATE: Tests	ATE_COV.1: Evidence of coverage ATE_FUN.1: Functional testing ATE_IND.2: Independent testing - sample
AVA: Vulnerability assessment	AVA_VAN.2: Vulnerability analysis

Table 5 *Security Assurance Requirements*

5.3 Security Requirements Rationale

5.3.1 Necessity and sufficiency analysis

SFR / TOE Security Objective	O.VIRUS	O.SELF_PROTECTION	O.INTERCEPTION	O.NOTIFY	O.UPDATE	O.RISK	O.AUDIT	O.IC_FEED	O.AUTHENTICATION	O.NONDISCLOSE	O.MANAGEMENT	O.EXPLOIT
FAU_GEN.1/INTERNAL	X	X	X	X		X		X				X
FAU_SAA.3/RISK		X	X			X						X
FAU_ARP.1	X	X		X		X						
FMT_MOF.1											X	
FAU_SAA.1/VIRUS	X											
FPT_ITT.1										X		
FAU_GEN.1/NDK				X			X					
FMT_SMF.1											X	
FMT_SMR.1											X	
FIA_UID.2									X			
FIA_UAU.2									X			
FAU_SAR.1/NDK				X			X					
FAU_SAR.3/NDK				X			X					
FPT_ITC.1						X		X				
FPT_ITI.1						X		X				
FDP_SDI.1					X							

SFR / TOE Security Objective	O.VIRUS	O.SELF_PROTECTION	O.INTERCEPTION	O.NOTIFY	O.UPDATE	O.RISK	O.AUDIT	O.IC_FEED	O.AUTHENTICATION	O.NONDISCLOSE	O.MANAGEMENT	O.EXPLOIT
FPT_TST.1		X										

Table 6 SFRs / TOE Security Objectives coverage

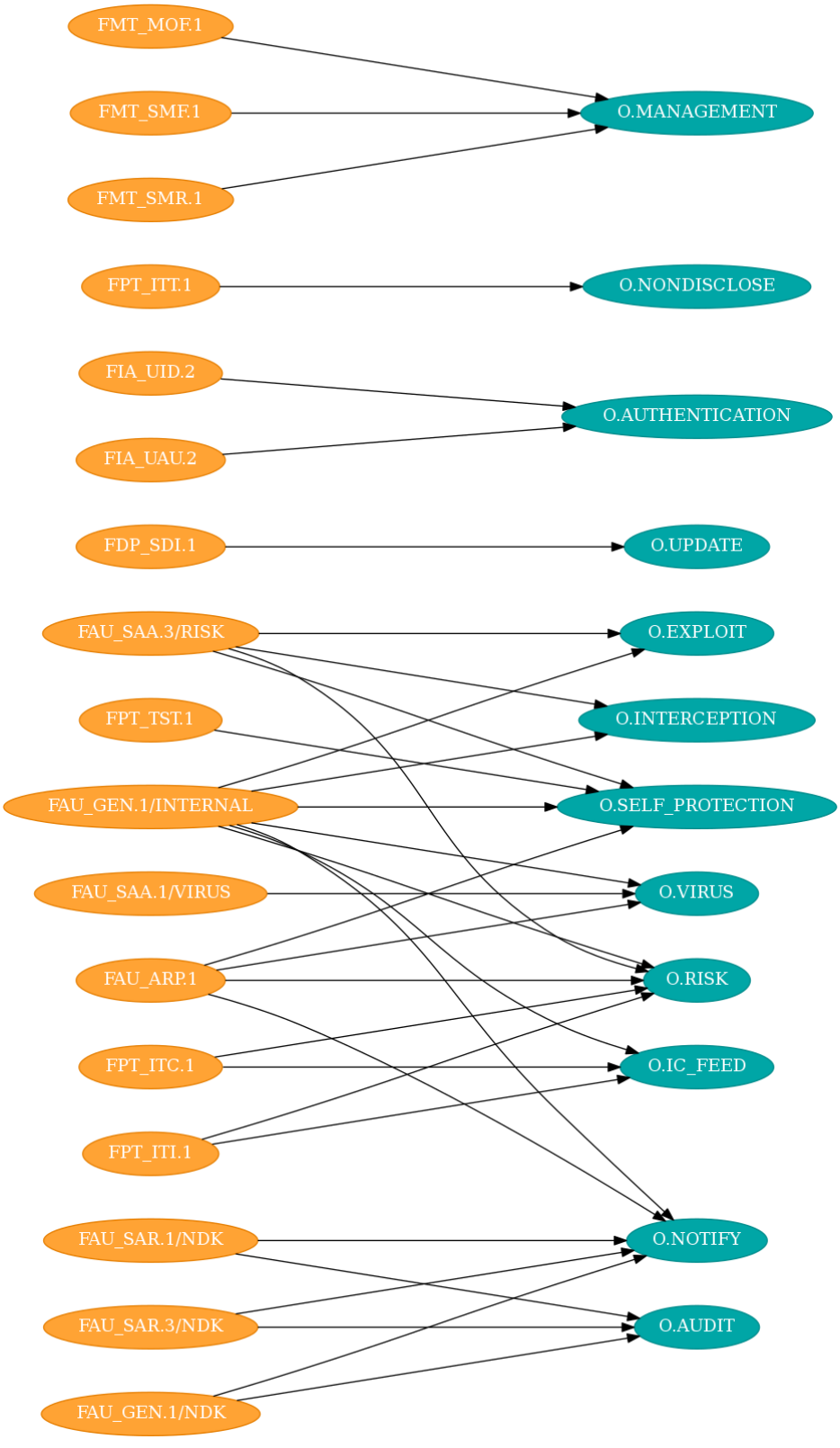


Figure 2 Mapping of SFRs to TOE Security Objectives

5.3.2 Security Requirement Sufficiency

O.VIRUS: The TOE maintains an internal database of process actions through **FAU_GEN.1/INTERNAL** . **FAU_SAA.1/VIRUS** allows detection of known viruses from the information stored in the database and the local database rules and blocking through **FAU_ARP.1**.

O.SELF_PROTECTION: This objective is implemented mainly as part of ADV_ARC.1 however other SFRs contribute through the audit (**FAU_GEN.1/INTERNAL**) analysis (**FAU_SAA.3/RISK**) and blocking of monitored process actions (**FAU_ARP.1**) and with the performance of self tests that verify the status of the TSF. (**FPT_TST.1**)

O.INTERCEPTION: The TOE maintains an internal database of process actions through **FAU_GEN.1/INTERNAL**. It will detect malicious sequence of actions through **FAU_SAA.3/RISK**.

O.NOTIFY: The TOE maintains an internal database of process actions through **FAU_GEN.1/INTERNAL**. When a malicious sequence of action is detected, it is notified to the suscribed NDKAPI processes (**FAU_ARP.1**). Other important events are logged through **FAU_GEN.1/NDK**, and they can be reviewed through the NDKAPI (**FAU_SAR.1/NDK** and **FAU_SAR.3/NDK**).

O.UPDATE: The TOE signatures and code can be updated. It will verify the integrity of updates through **FDP_SDI.1**.

O.RISK: This security objective is implemented through the audit (**FAU_GEN.1/INTERNAL**) analysis (**FAU_SAA.3/RISK**) and blocking of monitored process actions (**FAU_ARP.1**). FAU_SAA.3 uses for its decisions the information provided by Panda Collective Intelligence (**FPT_ITI.1** and **FPT_ITC.1**).

O.AUDIT: This security objective is implemented with the audit generation and review SFRs **FAU_GEN.1/NDK**, **FAU_SAR.1/NDK** and **FAU_SAR.3/NDK**.

O.IC_FEED: To contribute to Panda Collective Intelligence, the TOE sends the processes action stored through **FAU_GEN.1/INTERNAL** to the cloud, using a channel that provides confidentiality (**FPT_ITC.1**) and integrity (**FPT_ITI.1**).

O.AUTHENTICATION: NDKAPI client's identification and authentication is provided by **FIA_UID.2** and **FIA_UAU.2**.

O.NONDISCLOSE: **FPT_ITT.1** directly implements this security objective.

O.MANAGEMENT: There is only one role in the TOE, the used by the NDKAPI clients (**FMT_SMR.1**). It is used for the management functions that allows TOE configuration (**FMT_MOF.1** and **FMT_SMF.1**).

O.EXPLOIT: The TOE monitor process actions through **FAU_GEN.1/INTERNAL**. **FAU_SAA.3/RISK** allows detection and blocking of expected exploit techniques in commonly vulnerable processes.

5.3.3 SFR Dependency Rationale

5.3.3.1 Table of SFR dependencies

SFR	Required	Fulfilled	Missing
FAU_GEN.1/INTERNAL	FPT_STM.1	None	FPT_STM.1
FAU_SAA.3/RISK	None	None	None
FAU_ARP.1	FAU_SAA.1	FAU_SAA.1/VIRUS	None
FMT_MOF.1	FMT_SMR.1, FMT_SMF.1	FMT_SMR.1, FMT_SMF.1	None
FAU_SAA.1/VIRUS	FAU_GEN.1	FAU_GEN.1/INTERNAL	None
FPT_ITT.1	None	None	None
FAU_GEN.1/NDK	FPT_STM.1	None	FPT_STM.1
FMT_SMF.1	None	None	None
FMT_SMR.1	FIA_UID.1	FIA_UID.2 (hierarchically above FIA_UID.1)	None
FIA_UID.2	None	None	None
FIA_UAU.2	FIA_UID.1	FIA_UID.2 (hierarchically above FIA_UID.1)	None
FAU_SAR.1/NDK	FAU_GEN.1	FAU_GEN.1/NDK	None
FAU_SAR.3/NDK	FAU_SAR.1	FAU_SAR.1/NDK	None
FPT_ITC.1	None	None	None
FPT_ITI.1	None	None	None
FDP_SDI.1	None	None	None
FPT_TST.1	None	None	None

Table 7 SFR Dependencies

5.3.3.2 Justification for missing dependencies

FAU_GEN.1/INTERNAL dependency on FPT_STM.1

The dependency is resolved by the environment, which provides accurate time and date.

FAU_GEN.1/NDK dependency on FPT_STM.1

The dependency is resolved by the environment, which provides accurate time and date.

5.3.4 SAR Rationale

The SARs were chosen according to the market expected evaluation assurance level for the TOE type. ALC_FLR.1 was added to provide a tested flaw remediation procedure.

5.3.5 SAR Dependency Rationale

5.3.5.1 Table of SAR dependencies

SAR	Required	Fulfilled	Missing
ASE_CCL.1	ASE_INT.1, ASE_ECD.1, ASE_REQ.1	ASE_INT.1, ASE_ECD.1, ASE_REQ.2	None
ASE_ECD.1	None	None	None
ASE_INT.1	None	None	None
ASE_TSS.1	ASE_INT.1, ASE_REQ.1, ADV_FSP.1	ASE_INT.1, ASE_REQ.2, ADV_FSP.2	None
AGD_OPE.1	ADV_FSP.1	ADV_FSP.2	None
AGD_PRE.1	None	None	None
ASE_OBJ.2	ASE_SPD.1	ASE_SPD.1	None
ASE_REQ.2	ASE_OBJ.2, ASE_ECD.1	ASE_OBJ.2, ASE_ECD.1	None
ASE_SPD.1	None	None	None
ALC_CMC.2	ALC_CMS.1	ALC_CMS.2	None
ALC_CMS.2	None	None	None
ALC_DEL.1	None	None	None
ADV_ARC.1	ADV_FSP.1, ADV_TDS.1	ADV_FSP.2, ADV_TDS.1	None
ADV_FSP.2	ADV_TDS.1	ADV_TDS.1	None
ADV_TDS.1	ADV_FSP.2	ADV_FSP.2	None
ATE_COV.1	ADV_FSP.2, ATE_FUN.1	ADV_FSP.2, ATE_FUN.1	None
ATE_FUN.1	ATE_COV.1	ATE_COV.1	None
ATE_IND.2	ADV_FSP.2, AGD_OPE.1, AGD_PRE.1, ATE_COV.1, ATE_FUN.1	ADV_FSP.2, AGD_OPE.1, AGD_PRE.1, ATE_COV.1, ATE_FUN.1	None
AVA_VAN.2	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, AGD_OPE.1, AGD_PRE.1	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, AGD_OPE.1, AGD_PRE.1	None
ALC_FLR.1	None	None	None

Table 8 SAR dependencies

6 TOE Summary Specification

6.1 Class FAU: Security Audit

FAU_GEN.1/INTERNAL

These SFR describe one of the main functionality of the product, where processes actions are logged thanks to the system and user hooks placed in the operating system, that allow monitoring of every running process. Other interesting events like the one related with installation are also recorded.

This information cannot be reviewed through the NDKAPI, it is automatically sent to the cloud for the collective intelligence.

The following data is saved for each type of event:

Name	Explanation	Values
date	Date of the user's computer when the event was generated	Date
machineIP	IP address of the customer's computer	IP address
alertType	Category of the threat that triggered the alert	Malware PUP
executionStatus	The threat was run or not	Executed Not Executed
itemHash	Hash of the detected threat	String
itemName	Name of the detected threat	String
itemPath	Full path of the file that contains the threat	String

Table 9 Alert Table

Name	Explanation	Values
date	Date of the user's computer when the event was generated	Date
machineIP0	IP address of the customer's computer	IP address
machineIP1	IP address of an additional network card if installed	IP address
machineIP2	IP address of an additional network card if installed	IP address
operation	Operation performed	Install Uninstall Upgrade
osVersion	Operating system version	String
osServicePack	Service Pack version	String
osPlatform	Operating System platform	WIN32 WIN64

Table 10 *Install Table*

Name	Explanation	Values
date	Date of the user's computer when the event was generated	Date
machineIP	IP address of the customer's computer	IP address
user	Process user name	String
muid	Internal ID of the customer's computer	String in the following format xxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxxx
parentHash	Digest/hash of the file that accessed data	String
parentPath	Path of the process that accessed data	String
parentValidSig	Digitally signed process that accessed data	Boolean
parentCompany	Content of the Company attribute of the metadata of the file that accesses data	String
parentCat	Category of the file that accessed data	Goodware Malware PUP Unknown Monitoring

parentMWName	Malware name if the file that accessed data is classified as a threat	String Null if the item is not malware
childPath	Name of the data file accessed by the process. By default, only the file extension is indicated to preserve the privacy of the customer's data	String
loggedUser	User logged in on the computer at the time of file access	String

Table 11 *Monitoredopen Table*

Name	Explanation	Values
date	Date of the user's computer when the event was generated	Date
machineIP	IP address of the customer's computer	IP address
user	User name of the process that accessed or modified the	String

	registry	
muid	Internal ID of the customer's computer	String in the following format xxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxx
parentHash	Digest/hash of the file that accessed data	String
parentPath	Path of the process that accessed data	String
parentValidSig	Digitally signed process that accessed data	Boolean
parentCompany	Content of the Company attribute of the metadata of the file that accesses data	String
parentCat	Category of the file that accessed data	Goodware Malware PUP Unknown Monitoring
parentMWName	Malware name if the file that accessed data is classified as a threat	String Null if the item is not malware
regAction	Operation performed on the computer registry	CreateKey CreateValue ModifyValue

key	Affected registry branch or key	String
value	Name of the affected value under the registry key	String
valueData	Value content	String
loggedUser	User logged in on the computer at the time of registry access	String

Table 12 *MonitoredRegistry Table*

Name	Explanation	Values
date	Date of the user's computer when the event was generated	Date
machineIP	IP address of the customer's computer	IP address
user	Process user name	String
muid	Internal ID of the customer's computer	String in the following format xxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxxx
parentHash	Digest/hash of the file that accessed	String

	data	
parentPath	Path of the process that accessed data	String
parentValidSig	Digitally signed process that accessed data	Boolean
parentCompany	Content of the Company attribute of the metadata of the file that accesses data	String
parentCat	Category of the file that accessed data	Goodware Malware PUP Unknown Monitoring
parentMWName	Malware name if the file that accessed data is classified as a threat	String Null if the item is not malware
childHash	Child file digest/hash	String
childPath	Child process path	String
childValidSig	Digitally signed child process	Boolean
childCompany	Content of the company attribute of the child process metadata	String

childCat	Child process category	Goodware Malware PUP Unknown Monitoring
childMWName	Malware name if the child file is classified as a threat	String Null if the item is not malware

Table 13 *Notblocked Table*

Name	Explanation	Values
date	Date of the user's computer when the event was generated	Date
machineIP	IP address of the customer's computer	IP address
user	Process user name	String
op	Operation performed	CreateDir Exec CreatePE DeletePE LoadLib OpenCmp RenamePE CreateCmp
muid	Internal ID of the customer's computer	String in the following format

		xxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxx
parentHash	Digest/hash of the file that accessed data	String
parentPath	Path of the process that accessed data	String
parentValidSig	Digitally signed process that accessed data	Boolean
parentCompany	Content of the Company attribute of the metadata of the file that accesses data	String
parentCat	Category of the file that accessed data	Goodware Malware PUP Unknown Monitoring
parentMWName	Malware name if the file that accessed data is classified as a threat	String Null if the item is not malware
childHash	Child file digest/hash	String
childDriveType	Type of drive where the child process resides	Fixed Remote Removable
childPath	Child process path	String

childValidSig	Digitally signed child process	Boolean
childCompany	Content of the Company attribute of the child file metadata	String
childCat	Child file category	Goodware Malware PUP Unknown Monitoring
childMWName	Name of the malware found in the child file	String Null if the item is not malware
Ocs_Exec	Whether software considered as vulnerable was run or not	Boolean
Ocs_Name	Name of the software considered vulnerable	String
Ocs_Version	Version of the software considered vulnerable	String
clientCat	Item category in the agent cache	Goodware Malware PUP Unknown Monitoring
action	Action performed	Allow

		Block BlockTimeout
serviceLevel	Agent mode	Learning: The agent allows the execution of unknown processes Hardening: The agent prevents the execution of processes classified as threats Block: The agent prevents the execution of processes classified as threats and unknown processes

Table 14 *Ops Table*

Name	Explanation	Values
date	Date of the user's computer when the event was generated	Date
machineIP	IP address of the customer's computer	IP address
version	Version of the Adaptive Defense agent	String
user	Process user name	String
muid	Internal ID of the customer's computer	String in the following format

		xxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxx
hash	Digest/hash of the process	String
path	Program name and path	String
bytesSent	Number of bytes sent by the process since the last event was generated	Numeric
bytesReceived	Content of the Company attribute of the metadata of the file that accesses data	Numeric

Table 15 *ProcessNetBytes Table*

Name	Explanation	Values
date	Date of the user's computer when the event was generated	Date
machineIP	IP address of the customer's computer	IP address
version	Version of the Adaptive Defense	String

	agent	
user	User name of the process that modified the registry	String
op	Operation performed on the computer registry	ModifyExeKey CreateExeKey
hash	Digest/hash of the process that modified the registry	String
muid	Internal ID of the customer's computer	String in the following format xxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxx
targetPath	Path of the executable that the registry key points to	String
regKey	regKey	String
driveType	Type of drive where the process that accessed the registry resides	String
path	Path of the process that modified the registry	String
validSig	Digitally signed file that established	Boolean

	the connection	
company	Content of the Company attribute of the metadata of the file that established the connection	String
Cat	Process category	Goodware Malware PUP Unknown Monitoring
mwName	Malware name if the process is classified as a threat	String Null if the item is not malware

Table 16 *Registry Table*

Name	Explanation	Values
date	Date of the user's computer when the event was generated	Date
machineIP	IP address of the customer's computer	IP address
user	User name of the process that modified the registry	String
hash	Digest/hash of the process that	String

	modified the registry	
driveType	Type of drive where the process that accessed the registry resides	String
path	Path of the process that established the connection	String
protocol	Communications protocol used by the process	TCP UDP ICMP ICMPv6 IGMP RF
port	Communications port used by the process	0-65535
IP	Destination IP address	IP address
dstPort	Destination port	0-65535
dstIp6	IPv6 destination address	IP address
validSig	Digitally signed file that established the connection	Boolean
company	Content of the Company attribute of the metadata of the file that established the connection	String
Cat	Process category	Goodware

		Malware PUP Unknown Monitoring
mwName	Malware name if the process is classified as a threat	String Null if the item is not malware

Table 17 *Socket Table*

Name	Explanation	Values
date	Date of the user's computer when the event was generated	Date
machineIP	IP address of the customer's computer	IP address
user	Process user name	String
muid	Internal ID of the customer's computer	String in the following format xxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxx
parentHash	Digest/hash of the file that accessed data	String
parentPath	Path of the process that accessed data	String

parentValidSig	Digitally signed process that accessed data	Boolean
parentCompany	Content of the Company attribute of the metadata of the file that accesses data	String
parentCat	Category of the file that accessed data	Goodware Malware PUP Unknown Monitoring
parentMWName	Malware name if the file that accessed data is classified as a threat	String Null if the item is not malware
childHash	Child file digest/hash	String
childPath	Child process path	String
childValidSig	Digitally signed child process	Boolean
childCompany	Content of the Company attribute of the child file metadata	String
childCat	Child file category	Goodware Malware PUP Unknown Monitoring

clientCat	Item category in the agent cache	Goodware Malware PUP Unknown Monitoring
childMWName	Name of the malware found in the child file	String Null if the item is not malware
ToastResult	Result of the pop-up message	OK: The customer accepts the message Timeout: The pop-up message disappears due to non-action by the user Angry: The user rejects the block action Block Allow

Table 18 *Toast Table*

Name	Explanation	Values
date	Date of the user's computer when the event was generated	Date
machineIP	IP address of the customer's computer	IP address
user	Process user name	String
muid	Internal ID of the customer's	String in the following

	computer	format xxxxxxxx- xxxxxxxxxxxxxxxxxxxxxx
localCat	Item category from endpoint analysis	Goodware Malware PUP Unknown Monitoring
hash	Digest/hash of the process blocked	String
path	Path of the process blocked	String
ToastResult	Result of the pop-up message	OK: The customer accepts the message Timeout: The pop-up message disappears due to non-action by the user Angry: The user rejects the block action Block Allow

Table 19 *ToastBlocked Table*

Name	Explanation	Values
date	Date of the user's computer when the event was generated	Date
machineIP	IP address of the customer's computer	IP address

user	Process user name	String
muid	Internal ID of the customer's computer	String in the following format xxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxx
parentHash	Digest/hash of the file that accessed data	String
parentDriveType	Type of drive where the process that downloaded the file resides	Fixed Remote Removable
parentPath	Path of the process that accessed data	String
parentValidSig	Digitally signed process that accessed data	Boolean
parentCompany	Content of the Company attribute of the metadata of the file that accesses data	String
parentCat	Category of the file that accessed data	Goodware Malware PUP Unknown Monitoring
parentMWName	Malware name if the file that accessed data is classified as a	String Null if the item is not malware

	threat	
childHash	Child file digest/hash	String
childDriveType	Type of drive where the process that downloaded the file resides	Fixed Remote Removable
childPath	Child process path	String
childValidSig	Digitally signed child process	Boolean
childCompany	Content of the Company attribute of the child file metadata	String
childCat	Child file category	Goodware Malware PUP Unknown Monitoring
clientCat	Item category in the agent cache	Goodware Malware PUP Unknown Monitoring
childMWName	Name of the malware found in the child file	String Null if the item is not malware

Table 20 *URLdownload Table*

Name	Explanation	Values
date	Date of the user's computer when the event was generated	Date
machineIP	IP address of the customer's computer	IP address
criticalSoftEventType	Indicates the existence of vulnerable software	Present
itemHash	Digest of the vulnerable program found on the computer	String
fileName	Name of the vulnerable file	String
filePath	Full path of the vulnerable file	String
internalName	Content of the Name attribute of the vulnerable file metadata	String
companyName	Content of the Company attribute of the vulnerable file metadata	String
fileVersion	Content of the Version attribute of the vulnerable file	String

	metadata	
productVersion	Content of the ProductVersion attribute of the vulnerable file metadata	String

Table 21 *VulnerableAppsFound Table*

FAU_SAA.1/VIRUS **FAU_SAA.3/RISK**	This requirements describe the analysis capabilities of the TOE. For an easier understand of final consumers they have been divided in two. **FAU_SAA.1/VIRUS** describes the basic malware analysis capabilities, where the information recorded in **FAU_GEN.1/INTERNAL** is analysed to determine if a known virus or malware that matches the signatures managed by the TOE is present in the computer. **FAU_SAA.3/RISK** uses the more advanced malware analysis capabilities to determine, based on the process life, the configured operation mode and the Panda Collective Intelligence, if actions of a product may be blocked or not. This powerful security model is the main security feature of the TOE.
FAU_ARP.1	Given the analysis performed by **FAU_SAA.1/VIRUS** and **FAU_SAA.3/RISK**, the TOE will allow or deny the execution of a process or function. It may also send a notification to subscribed processes through the NDKAPI. Besides, known exploit techniques can be blocked before they are executed. The TOE also places hooks in vulnerable functions of well-known software to detect common exploit techniques and prevent its execution. This is the known as the anti-exploit feature of the TOE. Furthermore, the TOE can also block access to websites that are known for distributing malware or performing phishing attacks.

FAU_GEN.1/NDK **FAU_SAR.1/NDK** **FAU_SAR.3/NDK**	The TOE also generates audit data that can be reviewed by the client processes of the NDKAPI (Report information is saved encrypted in the Windows Event Log and can be reviewed) or send to subscribed clients of the NDK API (alert information). This audit data contains higher level information and it is expected that the final user will access it through the management console (out of the TOE scope) when it has been saved in the Management Cloud.

6.2 Class FDP: User Data Protection

FDP_SDI.1	Given the nature of malware, that is in constant evolution, the TOE has been designed to be highly updatable (both signatures and the product itself. The TOE will use MD5 to verify that the applicable updates don't have integrity problems.

6.3 Class FIA: Identification and Authentication

FIA_UAU.2 **FIA_UID.2**	Every client process action performed through the NDKAPI must be identified and authenticated before allowing its execution. The identification and authentication is implemented through the use of a signed token or through the validation of the signature of the process using the NDKAPI.

6.4 Class FMT: Security Management

FMT_MOF.1 **FMT_SMF.1** **FMT_SMR.1**	There is only one role in the TOE, the one used in every NDKAPI action. Some security management functions can be modified through the NDKAPI, like analysis exclusions for file or directories, whitelisting websites, or change the operation mode.

6.5 Class FPT: Protection of the TSF

FPT_ITI.1 **FPT_ITC.1**	The communication channel to the Collective Intelligence ensures integrity and confidentiality using the operating system provided functions to securely connect to a listening server.
FPT_TST.1	The TOE performs self test during startup and at regular intervals in order to verify that everything is working as expected
FPT_ITT.1	The communication between the client processes and the main process using the NDKAPI is considered an intra TOE communication, given that the DLL file that implements the NDKAPI client is part of the TOE and is loaded in the client process. This communication is performed through system pipes and provides disclosure protection through the use of a secret encryption key.

7 Acronyms

The following table shows the acronyms used in this Security Target

Acronym	Meaning
ST	Security Target
PP	Protection Profile
CC	Common Criteria
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSFi	TSF Interface
IT	Information Technology
OSP	Organisational Security Policies
EAL	Evaluation Assurance Level
AV	AntiVirus
API	Application Programming Interface
NDK	Nano Development Kit
IoC	Inversion of Control
DI	Dependency Injection
COM	Component Object Model
IPC	Interprocess communication
AES	Advanced Encryption Standard
RPC	Remote procedure call
EDR	Endpoint Detection and Response
IDS	Intrusion Detection System
OS	Operating System
TSC	TSF Scope of Control
TSS	TOE Summary Specification
KRE	Kernel Rules Enforcement

Table 22 Abbreviations

8 Glossary of Terms

Term	Meaning
Augmentation	Addition of one or more requirement(s) to a package
Evaluation Assurance Level	Set of assurance requirements drawn from CC Part 3, representing a point on the CC predefined assurance scale, that form an assurance package
Operational Environment	Environment in which the TOE is operated
Protection Profile	Implementation-independent statement of security needs for a TOE type
Security Target	Implementation-dependent statement of security needs for a specific identified TOE
Target Of Evaluation	Set of software, firmware and/or hardware possibly accompanied by guidance

Table 23 Glossary of terms

9 Document References

The following table shows the documents referenced in this Security Target

Reference	Document
CC31R4P1	Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 4, Part 1: Introduction and general model
CC31R4P2	Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 4, Part 2: Security functional components
CC31R4P3	Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 4, Part 3: Security assurance components
CEM31R4	Common Criteria Evaluation methodology, Version 3.1, Revision 4
AD360-GUIDE	Adaptive Defense 360 Guide for network administrators v2.3.5
NDKAPI	NDK API Latest version

Table 24 List of document references