
# Destroy 2.01
# and
# Destroy Lite 2.01

## Security Target Version 1.18

## July 2003

# Acknowledgments

This Security Target was jointly prepared by David Quail and Sue Cramer-Roberts of The Australian Software Company and Anne Robins and Peter Lilley of 90 East.

This document was evaluated by Bruce Legge of LogicaCMG.

Destroy and Destroy Lite 2.01 are products developed by The Australian Software Company.

www.destroy.com.au

# Table of Contents

# List of Tables

# Conventions and Terminology

## Conventions

The notation, formatting, and conventions used in this Security Target are consistent with those used in Version 2.1 of the Common Criteria [CC]. Selected presentation choices are discussed here to aid the Security Target reader. The CC allows several operations to be performed on functional and assurance requirements: The allowable operations defined in paragraph 2.1.4 of Part 2 of the CC [CC2] are *refinement, selection, assignment* and *iteration*.

- The assignment operation is used to assign a specific value to an unspecified parameter, such as the length of a password. An assignment operation is indicated by showing the value in square brackets, i.e. [assignment_value(s)].
- The refinement operation is used to add detail to a requirement, and thus further restricts a requirement. Refinement of security requirements is denoted by **bold text**.
- The selection operation is picking one or more items from a list in order to narrow the scope of a component element. Selections are denoted by *underlined italicised* text.
- Iterated functional and assurance requirements are given unique identifiers by appending to the base requirement identifier from the CC an iteration number inside parenthesis, for example, FMT_MTD.1.1 (1) and FMT_MTD.1.1 (2) refer to separate instances of the FMT_MTD.1 security functional requirement component.

All operations described above are used in this Security Target. *Italicised text* is used for both official document titles and text meant to be emphasised more than plain text.

## Terminology

The terminology used in the Security Target is that defined in the Common Criteria [CC1, CC2]. The following additional TOE specific terminology is included to assist the consumer of the Security Target:

| | |
|---|---|
| **ACSI 33** | Australian Communications-Electronic Security Instruction 33<br>ACSI 33 is a DSD publication providing guidance to Australian Government agencies on the protection of their information. Handbook 6, Media Security Para 605 specifically applies for this Security Target. |
| **BIOS** | The BIOS is built-in software that determines what a computer can do without accessing programs from a disk. On PCs, the BIOS contains code required to control the keyboard, display screen, disk drives, serial communications, and a number of miscellaneous functions. The PC BIOS is fairly standardized, so all PCs are functionally similar at this level (although there are different BIOS versions). |

| | |
|---|---|
| **Cylinder** | A single track location on all the platters making up a hard disk. For example, if a hard disk has four platters, each with 600 tracks, then there will be 600 cylinders, and each cylinder will consist of 8 tracks (assuming that each platter has tracks on both sides). |
| **Destroy 2.01** | The Target of Evaluation.<br><br>A software program developed by The Australian Software Company that securely removes data from hard drives in a way that makes data recovery effectively impossible. Destroy 2.01 uses multiple disk reads and writes of various data values to ensure that no residual data remains on a hard drive. |
| **Destroy Lite 2.01** | The Target of Evaluation<br><br>A software program developed by The Australian Software Company that securely and quickly removes data from hard drives in a way that makes data recovery effectively impossible. Destroy Lite 2.01 is faster than Destroy 2.01 because it uses fewer read and write operations and hence may be more suitable for sanitising less sensitive data than Destroy 2.01. |
| **Destroy Operators** | The Destroy product does not have the concept of administrators and users, rather a Destroy Operator is a person authorised to run the software. |
| **DVC** | Destroy Validation Check<br><br>Utility for validating the integrity of the Destroy executables. |
| **Destroy Validation Key** | The MD5 hash value used to confirm the integrity of the Destroy executable prior to use. The Destroy Validation Key is provided both in printed form and electronically in the DVC integrity checking utility. |
| **DSD** | Defence Signals Directorate<br><br>DSD is Australia's national authority for information security. |
| **Head** | The mechanism that reads data from or writes data to a magnetic disk. The head is sometimes called a read/write head. Hard disk drives have many heads, usually two for each platter. |
| **Host Protected Area** | The Host Protected Area (HPA) is a feature of IDE hard disks that conform to the ATA-4 hard disk standard which is the ANSI standard that details the communication of a hard disk with the computer. The Host Protected Area is a reserved area for data storage outside the normal operating system file system and is both read and write protected. |
| **Resource** | Within the context of this Security Target, a resource is a hard drive. |
| **Sanitisation** | The process of erasing as far as is possible the information from the media or equipment. The process of sanitisation does not automatically change the classification of the media or equipment. Note that sanitisation does not involve destroying the media or equipment. |
| **Sector** | The smallest unit that can be accessed on a disk. When a disk undergoes a low-level format, it is divided into tracks and sectors. The tracks are concentric circles around the disk and the sectors are segments within each circle.<br><br>A sector that cannot be used due to a physical flaw on the disk is called a bad sector. |

| | |
|---|---|
| **Track** | A ring on a disk where data can be written. For hard disks, each platter is divided into tracks, and a single-track location that cuts through all platters (and both sides of each platter) is called a cylinder. Hard disks many thousands of cylinders. Each track is further divided into a number of sectors. |

## References


[CC] Common Criteria for Information Technology Security Evaluation, Version 2.1, August 1999.

[CC1] Common Criteria Part 1: Introduction and General Model, Version 2.1, CCIMB-99-031, August 1999.

[CC2] Common Criteria Part 2: Security Functional Requirements, Version 2.1, CCIMB-99-032, August 1999.

[CC3] Common Criteria Part 3: Security Assurance Requirements, Version 2.1, CCIMB-99-033, August 1999.

[CEM] Common Evaluation Methodology Part 2: Evaluation Methodology, Version 1.0, CEM-99/045, August 1999.


# Document Organisation

**Section 1** provides the introductory material for the Security Target.

**Section 2** provides general purpose and TOE description.

**Section 3** provides a discussion of the expected environment for the TOE. This section also defines the set of threats that are to be addressed by either the technical countermeasures implemented in the TOE hardware or software or through the environmental controls.

**Section 4** defines the security objectives for both the TOE and the TOE environment.

**Section 5** contains the functional and assurance requirements derived from the Common Criteria, Part 2 and 3 [CC2, CC3], respectively that must be satisfied by the TOE.

**Section 6** identifies the IT security functions provided by the TOE and also identifies the assurance measures targeted to meet the assurance requirements.

**Section 7** makes any protection profile claims applicable to the TOE.

**Section 8** provides a rationale to explicitly demonstrate that the information technology security objectives satisfy the policies and threats. Arguments are provided for the coverage of each policy and threat. The section then explains how the set of requirements are complete relative to the objectives, and that each security objective is addressed by one or more component requirements. Arguments are provided for the coverage of each objective. Next, Section 8 provides a set of arguments that address dependency analysis,

strength of function issues, and the internal consistency and mutual supportiveness of the security target requirements.

**Appendix A** documents an acronym list to define frequently used acronyms applicable to the TOE.

# 1 Introduction

This introductory section presents *security target (ST)* identification information and an overview of the ST. A statement of Common Criteria conformance is also provided.

## 1.1 ST and TOE Identification

This section provides information needed to identify and control this ST and its Target of Evaluation (TOE). This ST targets an **Evaluation Assurance Level (EAL) 2 augmented** level of assurance for the TOE.

| | |
|---|---|
| **ST Title:** | Destroy and Destroy Lite Security Target Version 1.18 |
| **TOE Identification:** | Destroy 2.01 and Destroy Lite 2.01 |
| **CC Version:** | Common Criteria for Information Technology Security Evaluation, Version 2.1 Final |
| **ST Evaluation:** | Australasian Information Security Evaluation Program, Defence Signals Directorate, Australian Department of Defence |
| **Author(s):** | Anne Robins, Peter Lilley, David Quail |
| **Keywords:** | Media Security, Disk Sanitisation, Residual Information Protection |

## 1.2 Security Target Overview

**Product Background**

The Australian Software Company (TASC) have developed a suite of software programs that securely removes all data from PC hard disks in a way that makes data recovery practically impossible. The Target of Evaluation (TOE) comprises the product *Destroy 2.01* and the product *Destroy Lite 2.01*. TASC commenced development of this product in mid 1999 in consultation with the Australian Government.

### Market for Product

The market for the Destroy products was identified because of the problems faced by a large number of government agencies and corporate organisations who had no way of securely removing data from computer hard disks at the end of their lease cycle, nor when recycling machines internally, and often a number of machines were found to have sensitive information stored on them that could have been compromised. In some instance agencies were returning desktop and notebook computers without hard disks and were physically damaging hard drives as a means of protecting their data. Finance companies' lease terms also often state that all computers are to be returned in the same condition as when initially leased. Therefore computers returned without hard disks or damaged hard disks, have to be replaced with a disk of the same size, or be upgraded where no similar replacement disk is available.

### The Destroy Products

The development work of TASC, in consultation with the Australian government has resulted in the production of the two Destroy products – Destroy 2.01 and Destroy Lite 2.01. The Destroy 2.01 product provides a higher level of confidence in the removal of residual data than Destroy Lite 2.01, due to the more extensive sanitisation process used. Conversely, the Destroy Lite 2.01 product will complete disk sanitisation operations more quickly. Potential purchasers should consider both their security needs and their cost and performance requirements when selecting the appropriate product.

The Destroy products securely sanitise PC computer hard disks, are effectively independent of disk size or operating system, and comply with the requirements of the Australian Communications-Electronic Security Instructions 33 (ACSI 33), Handbook 6, Media Security, paragraph 605.

The Destroy products completely overwrite all hard disks with hexadecimal value 00 and hexadecimal value FF. This process is repeated three times for Destroy (and only once for Destroy Lite) and then a final write process overwrites a hard disk with random ASCII characters. Following each complete write process the program verifies, by reading each sector from the hard drive, to ensure data has been overwritten. The final random write process is not verified.

Any errors in attempting to read or write to a sector will be reported and recorded. At the completion of the disk sanitisation, the Destroy Operator is presented with a summary of the process including notification of any errors and a statement of the overall outcome (success or failure) of the process.

A utility, forming part of the product, provides the ability to conduct integrity checks of the Destroy and Destroy Lite programs to ensure that the programs have not been changed prior to use. This utility is termed the Destroy Validation Check (DVC).

## 1.3 Common Criteria Conformance

The TOE is conformant with Part 2 of the CC, version 2.1 [CC2] and the assurance requirements of EAL 2 augmented with ADV_SPM.1, as defined in Part 3 of the CC, version 2.1 [CC3].

# 2 TOE Description

This section provides context for the TOE evaluation by identifying the product type and describing the evaluated configuration.

## 2.1 Overview of the Product (TOE)

The Destroy products (Destroy 2.01 and Destroy Lite 2.01) are PC-based BIOS-compatible software applications, which utilise low level (BIOS) read and write functions to overwrite all hard disk data.

On start-up Destroy interrogates the BIOS to determine the reported number of drives and the capacity of each hard drive. Destroy then physically checks each hard disk, using the Destroy BIOS Query Function, to determine the actual physical capacity, which may differ from the BIOS settings. The configuration of the hard drive(s) in the Bios does not need to match the actual drive parameters. The only requirement is that the BIOS is configured to know that all hard disks are installed. Destroy then proceeds to overwrite the entire drive based on the actual physical capacity determined on start-up. This will include any hidden tracks (cylinders) found on a hard disk by Destroy.

Destroy will process variable sector sizes, variable track values and variable head values.

The Destroy Operator will be able to view the following information once the program starts up:

Destroy Version 2.01 (or Destroy Lite Version 2.01)
Copyright (c) 1999-2002 The Australian Software Company Pty Limited.
All rights reserved

Write 0x00 and 0xFF *w* times, and 1 Random Write – Total of *x* passes
Hard Disk Information Detected for *y* disks

| | BIOS Reported | | | | | Destroy Actual | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | C | H | S | Sectors | Size MB | C | H | S | Sectors | Size MB |
| Disk *z* | [a] | [b] | [c] | [d] | [e] | [f] | [g] | [h] | [i] | [j] |

The Destroy products completely overwrite all hard disks with hexadecimal value zero (ASCII Character 0) and hexadecimal value FF (ASCII character 255). This process is repeated three times for Destroy 2.01 (and once for Destroy Lite 2.01) and then a final write process overwrites the hard disk(s) with random ASCII characters between 0 and 255.

Following each complete write process (i.e. all tracks are written first), the program verifies by reading each sector from the hard drive to ensure all data has been overwritten. The program does not verify the random process.

The Destroy products take advantage of the BIOS track-by-track reading and writing facility to provide high levels of performance. If a read or write failure occurs, the product will reset the hard disk controller and attempt the read or write again. This will occur four times, and if it continues to fail then the sector will be marked as bad.

The program will attempt to read or write to a bad sector three times, and if unsuccessful increments the error count. An error count greater than zero will generate a failure message at the end of the entire sanitisation process. A read or write failure where a controller reset corrects the problem is not consider a failure.

When the product has finished running, the Destroy Operator will be able to view the following report on screen:

Destroy Version 2.01 (or Destroy Lite Version 2.01)
Copyright (c) 1999-2002 The Australian Software Company Pty Limited
All Rights Reserved

| | Error Count | | | | | |
|---|---|---|---|---|---|---|
| Results | Write | Read | Verify | Drive Details | | |
| Zero | [a] | [b] | [c] | Drive | : | [u] C: [v] H: [w] S: [x] |
| One | [d] | [e] | [f] | Sectors | : | [y] ([z] MB) |
| Random | [g] | [h] | [i] | Time | : | [dd/mm/yy] [hh:mm] – [dd/mm/yy] [hh:mm] |

- [u] is the drive being reported
- [y] is the total number of sectors
- [z] is the total size of the hard disk in megabytes (Mb), where 1Mb equals 1024 Kb (kilobytes)
- [dd/mm/yy] and [hh:mm] show the day, month, year, and times for the start and finish of the Destroy process.

For computers with more than one hard disk, Destroy will display this information in succession for each hard disk processed. At the end of the entire process a final overall outcome of the process will be displayed as follows:

> DESTROY successfully processed [x] drive(s) (or DESTROY Lite successfully processed [x] drive(s)).
> **OR**
> DESTROY did NOT complete successfully, errors in drive(s) [x] (or DESTROY Lite did NOT complete successfully, errors in drive(s) [x]).

The disk wipe success is dependent on the number of errors found on a hard disk. Destroy 2.01 and Destroy Lite 2.01 will only fail if a sector/track cannot be read or written to three times, however, the program will reset itself and continues to wipe until the entire drive is completed.

The Destroy programs do not write to the floppy disk drive, or to any other types of storage media including tape drives, CD-ROM drives or Zip drives. The programs are executed from a write protected bootable floppy disk for protection against tampering. Destroy and Destroy Lite may also run off a bootable CD-ROM, however TASC do not provide Destroy or Destroy Lite on CD-Rom unless it is specifically requested by the client. It is possible to copy Destroy and Destroy Lite onto a bootable CD Rom although you will need to contact your IT Department for instructions on how to create a bootable CD Rom. The Destroy programs accept only one parameter, "/R" which forces the program to start without requiring user confirmation. This is used when requiring an automatic bootable Destroy process. Thus the Destroy operator is only required to insert the disk into the bootable floppy drive and switch the computer on or reboot and the program will commence. If the "/R" parameter is not used, then the Destroy Operator will be asked to confirm if they want to proceed with the sanitisation operation before the Destroy program proceeds.

Since the booting process forces the machine to utilise the operating system provided on the bootable disk (for Destroy 2.01 and Destroy Lite 2.01 the operating system is DOS 7 also known as Windows 98 DOS), it will allow the Destroy programs to overwrite all hard drives regardless of the operating system. The program overwrites the File Allocation Table as well as directory structure.
A MS-Windows based utility (Windows 95 or higher or Windows NT) is a component of both the Destroy 2.01 and Destroy Lite 2.01 products, and is provided to enable an integrity check of the Destroy or Destroy Lite program to ensure it has not been changed prior to use. This utility, called "DVC", has an inbuilt MD5 hash value, termed the Destroy Validation Key, that is compared with the hash of the Destroy program calculated by the utility. A printed copy of the Destroy Validation Key is also provided. If the calculated hash is different to the Destroy Validation Key then an error message is displayed. Additionally, the calculated hash of the Destroy program is displayed on the screen to allow the Destroy Operator to perform a manual match to ensure that the integrity checking utility has not been altered.

Destroy complies with the requirements defined in ACSI 33, Handbook 6, Media Security for the sanitisation of media used by the Australian Government.

## 2.2 Physical Scope of the TOE

The physical scope of the TOE includes the hardware and software elements identified in Table 1.

**Table 1: Physical Components of the TOE**

| Physical TOE Components | Hardware/Software Platforms |
|---|---|
| **Destroy 2.01**<br>**or**<br>**Destroy Lite 2.01** | DOS 7 (also known as Windows 98 DOS) Bootable write protected floppy disk containing the Destroy or Destroy Lite executable.<br>Hardware requirements:<br>• PC containing the hard drive(s) for sanitisation and capable of running the DOS 7 (also known as Windows 98 DOS) bootable disk;<br>• A floppy disk drive suitable for the supplied media type. |
| **"DVC" Integrity Checking Program** | Floppy disk containing an install file that installs a single executable.<br>Operating system requirements:<br>• Windows 95 or higher;<br>• Windows NT 4.0.<br>Hardware requirements:<br>• PC capable of running above operating systems; and<br>• A floppy disk drive suitable for the supplied media type. |

## 2.3 Security Features

The TOE provides the following security features:

**Table 2: Summary of TOE Security Features**

| Feature | Description |
|---|---|
| **Sanitisation of Hard Drives** | The TOE completely overwrites each hard disk with hexadecimal value 00 and hexadecimal value FF. This process is repeated three times for Destroy (and only once for Destroy Lite) and then a final write process overwrites the hard disks with random ASCII characters. Following each complete write process the program verifies, by reading each sector from the hard drive, to ensure data has been overwritten.<br>If a failure occurs the TOE will reset the hard disk controller and attempt the read/write again. This will occur four times and if it continues to fail then the sector will be marked as bad. The program will attempt to read or write to a bad sector three times and if unsuccessful, increments an error count. An error count greater than zero will generate a failure condition. |
| **Integrity Self Checking** | The TOE includes a utility to verify the integrity of the TOE executable. |
| **Sanitisation Reporting** | During the sanitisation process the TOE displays errors on the screen as they are encountered. The TOE also displays a status at the end of the sanitisation process. |

# 3 TOE Security Environment

In order to clarify the nature of the security problem that the TOE is intended to solve, this section describes the following:

- Any assumptions about the security aspects of the environment and/or of the manner in which the TOE is intended to be used.
- Any known or assumed threats to the assets against which specific protection within the TOE or its environment is required.
- Any organisational security policy statements or rules with which the TOE must comply.

## 3.1 Secure Usage Assumptions

The following assumptions relating to the operation of the TOE are made.

**Table 3: Assumptions**

| Name | Description |
|---|---|
| **A.ADMIN_DOCS** | The Destroy User will follow all policies and procedures defined in the Destroy documentation to ensure the secure usage of Destroy. |
| **A.NO_EVIL** | Destroy Users are assumed to be non-hostile and are trusted to perform their duties in a competent manner. |
| **A.ATTACK** | Destroy will be used to prevent unauthorised access to stored data and potential attackers are assumed to have a medium level of expertise and resources and a medium level of motivation. |
| **A.NO_SOFTWARE** | It is assumed that while the Destroy program is in operation, no other software will be active. |
| **A.NO_NETWORK** | It is assumed that while the Destroy program is in operation, no network connectivity is active for the computer containing the target hard drive(s). |
| **A.BOOTABLE** | The Destroy program will be run from a write-protected, bootable device such as a floppy disk. |
| **A.BIOS_CONFIG** | The BIOS is configured to know that all hard disks are installed. For IDE hard disks that support the Host Protected Area (HPA) feature the HPA has been reset to the manufacturer's specification or has been removed. |

## 3.2 Threats to Security

Threats may be addressed either by the TOE or by its intended environment (for example, using personnel, physical, or administrative safeguards). These two classes of threats are discussed separately.

### 3.2.1 Threats Addressed by the TOE

The TOE addresses the following threats.

**Table 4: Threats**

| Name | Description |
|---|---|
| **T.DATA_ACCESS** | An unauthorised person attempts to access sensitive data stored on a hard drive that has been redeployed, transferred out of the organisation's control, or discarded. |
| **T.DATA_DELETED** | An unauthorised person attempts to recover sensitive data remaining after the data has been deleted from a hard drive that has been redeployed, transferred out of the organisation's control, or discarded. |
| **T.DATA_FORMAT** | An unauthorised person attempts to recover sensitive data remaining after formatting of a hard drive that has been redeployed, transferred out of the organisation's control, or discarded. |
| **T.DATA_HIDDEN** | An unauthorised person attempts to recover sensitive data from tracks or sectors of a hard drive that are outside of the defined configuration parameters for that hard drive and that has been redeployed, transferred out of the organisation's control, or discarded. |
| **T.MODIFY** | An unauthorised person attempts to modify the software comprising the TOE to circumvent or disable its security features. |
| **T.UNAWARE** | An authorised user of the TOE executes the software but is unaware of a failure to completely sanitise a hard drive. |

### 3.2.2 Threats Addressed by the Operating Environment

The TOE Operating Environment is not required to explicitly address any threats, although the TOE Operating Environment is constrained by the assumptions made above in Table 3.

## 3.3 Organisational Security Policies

The table following describes the organisational security policies relevant to the operation of the TOE.

**Table 5: Organisational Security Policies**

| Name | Description |
|---|---|
| **P.DISPOSAL** | An organisation using the TOE must define an appropriate policy for the identification, disposal and sanitisation of hard drives. If the organisation is within the Australian government then the policy should be consistent with the guidelines in ACSI 33. |
| **P.INTEGRITY_CHECK** | An organisation using the TOE must define an appropriate policy defining how often and under what circumstances the integrity of the TOE is to be checked. |

# 4 Security Objectives

The security objectives are a concise statement of the intended response to the security problem. These objectives indicate, at a high level, how the security problem, as characterised in the "Security Environment" section of the ST, is to be addressed. Just as some threats are to be addressed by the TOE and others by its intended environment, so some security objectives are for the TOE and others are for its environment. These two classes of security objectives are discussed separately.

## 4.1 Security Objectives for the TOE

The security objectives for the TOE are as described in the following table.

**Table 6: Security Objectives for the TOE**

| Name | Description |
|---|---|
| **O.SANITISE** | The TOE shall prevent unauthorised access to data stored on a hard drive that is to be redeployed, transferred out of the organisation's control, or discarded. |
| **O.INTEGRITY** | The TOE shall provide a means of detecting loss of integrity affecting operation. |
| **O.NOTIFY** | The TOE shall provide a means of notifying authorised users of the success and/or failure to sanitise a hard drive that is to be redeployed, transferred out of the organisation's control, or discarded. |

## 4.2 Security Objectives for the Environment

The security objectives for the TOE environment are those specified in the table below.

**Table 7: Security Objectives for the Environment**

| Name | Description |
|---|---|
| **OE.OPERATE** | Those responsible for the TOE shall ensure that the TOE is installed, managed and operated in a manner consistent with defined organisational policies and the TOE documentation. In addition, those responsible for the security of the organisation shall ensure that all appropriate background checks, psychological assessments, and security clearances, as required, are conducted for all Destroy Operators. |
| **OE.MEDIA_ACCESS** | Those responsible for the TOE shall ensure that access to the media containing the TOE executable is controlled in a manner consistent with defined organisational policies. |
| **OE.BIOS** | Those responsible for the TOE shall ensure that the BIOS of the machine containing the hard drives to be sanitised has been correctly configured to know that all hard disks are installed. Those responsible for the TOE shall also ensure that Host Protected Area (HPA) has been reset to the manufacturer's specification or removed. |

# 5 IT Security Requirements

## 5.1 TOE Security Functional Requirements

This section contains the functional requirements for the TOE. The functional requirements are listed in summary form in Table 8, below.

**Table 8: TOE Security Functional Requirements**

| No. | Component | Component Name |
|---|---|---|
| **Class FAU: Audit** | | |
| 1 | FAU_ARP.1 | Security alarms |
| 2 | FAU_GEN.1 | Audit data generation |
| 3 | FAU_SAA.1 | Potential violation analysis |
| **Class FCS: Cryptographic Support** | | |
| 4 | FCS_COP.1 | Cryptographic Operation |
| **Class FDP: User data protection** | | |
| 5 | FDP_RIP.2 | Full residual information protection |
| **Class FPT: Protection of the TSF** | | |
| 6 | FPT_AMT.1 | Abstract machine testing |
| 7 | FPT_PHP.1 | Passive detection of physical attack |
| 8 | FPT_RCV.4 | Function recovery |
| 9 | FPT_TST.1 | TSF testing |

The following sections contain the functional components from the Common Criteria Part 2 [CC2] (CC) with the operations completed. The standard CC text is in regular font; the text inserted by the Security Target (ST) author is in accordance with the conventions described in at the beginning of this document.

### 5.1.1 Audit (FAU)

#### Security alarms (FAU_ARP.1)

Hierarchical to: No other components.

**FAU_ARP.1.1** The TSF shall take [the action to write to the display] upon detection of a potential security violation.

Dependencies: FAU_SAA.1 Potential violation analysis

#### Audit data generation (FAU_GEN.1)

Hierarchical to: No other components.

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

a) Startup and shutdown of the audit functions;

b) All auditable events for the *not specified* level of audit; and

c) [process completed successfully;

process did not complete successfully;]

**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 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 ST,

[start time;

finish time;

total number of cylinders;

total number of heads;

total number of sectors;

error count for 0x0 write/retry/read/verify;

error count for 0xFF write/retry/read/verify;

error count for random write;

success or failure of the sanitisation process;

the number of drives; and

the error status of the drive.].

Dependencies: FPT_STM.1 Reliable time stamps

### Potential violation analysis (FAU_SAA.1)

| | |
|---|---|
| Hierarchical to: | No other components. |
| **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 TSP. |
| **FAU_SAA.1.2** | The TSF shall enforce the following rules for monitoring audited events:<br>a) Accumulation or combination of [write/retry/read/verify errors] known to indicate a potential security violation;<br>b) [none]. |
| Dependencies: | FAU_GEN.1 Audit data generation |

## 5.1.2 Cryptographic support (FCS)

### Cryptographic operation (FCS_COP.1)

| | |
|---|---|
| Hierarchical to: | No other components. |
| **FCS_COP.1.1** | The TSF shall perform [calculation and verification of cryptographic hashes] in accordance with a specified cryptographic algorithm [MD5] and cryptographic key sizes [n/a], that meet the following: [requirements for cryptography as defined by RFC 1321]. |
| Dependencies | [FDP_ITC.1 Import of user data without security attributes<br>or<br>FCS_CKM.1 Cryptographic key generation]<br>FCS_CKM.4 Cryptographic key destruction<br>FMT_MSA.2 Secure security attributes |

## 5.1.3 User data protection (FDP)

### Full residual information protection (FDP_RIP.2)

| | |
|---|---|
| Hierarchical to: | FDP_RIP.1 |
| **FDP_RIP.2.1** | The TSF shall ensure that any previous information content of a resource is made unavailable upon the *deallocation of the resource from* all objects. |
| Dependencies: | No dependencies |

## 5.1.4 Protection of the TSF (FPT)

### Abstract Machine Testing (FPT_AMT.1)

| | |
|---|---|
| Hierarchical to: | No other components |
| **FPT_AMT.1.1** | The TST shall run a suit of tests *during initial start-up, periodically during normal operation* to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the TSF. |
| Dependencies: | No dependencies |

### Passive detection of physical attack (FPT_PHP.1)

| | |
|---|---|
| Hierarchical to: | No other components |
| **FPT_PHP.1.1** | The TSF shall provide unambiguous detection of physical tampering that might compromise the TSF. |
| **FPT_PHP.1.2** | The TSF shall provide the capability to determine whether physical tampering with the TSF's devices or TSF's elements has occurred. |
| Dependencies: | FMT_MOF.1 Management of security functions behaviour |

### Function recovery (FPT_RCV.4)

| | |
|---|---|
| Hierarchical to: | No other components |
| **FPT_RCV.4.1** | The TSF shall ensure that [the security functions (SF) WRITE_K, WRITE_R, and VALID_K and the scenario of one failure, and the SFs WRITE_K, WRITE_R and VALID_K and the scenario of three successive failures] have the property that the SF either completes successfully, or for the indicated failure scenarios recovers to a consistent and secure state. |
| Dependencies: | ADV_SPM.1 Informal TOE security policy model |

### TSF testing (FPT_TST.1)

| | |
|---|---|
| Hierarchical to: | No other components |
| **FPT_TST.1.1** | The TSF shall run a suite of self tests *at the conditions* [following the execution of the WRITE_K function] to demonstrate the correct operation of the TSF. |
| **FPT_TST.1.2** | The TSF shall provide authorised users with the capability to verify the integrity of TSF data. |
| **FPT_TST.1.3** | The TSF shall provide authorised users with the capability to verify the integrity of stored TSF executable code. |
| Dependencies: | FPT_AMT.1 Abstract machine testing |

## 5.2 TOE Security Assurance Requirements

This section contains the assurance requirements for the TOE. The assurance requirements are listed in summary form in Table 9: TOE Security Assurance Requirements, below.

**Table 9: TOE Security Assurance Requirements**

| No. | Component | Component Name |
|---|---|---|
| **Class ACM: Configuration management** | | |
| 1 | ACM_CAP.2 | Configuration items |
| **Class ADO: Delivery and Operation** | | |
| 2 | ADO_DEL.1 | Delivery procedures |
| 3 | ADO_IGS.1 | Installation, generation and start-up procedures |
| **Class ADV: Development** | | |
| 4 | ADV_FSP.1 | Informal functional specification |
| 5 | ADV_HLD.1 | Descriptive high level design |
| 6 | ADV_RCR.1 | Informal correspondence demonstration |
| 7 | ADV_SPM.1 | Informal TOE security policy model |
| **Class AGD: Guidance documents** | | |
| 8 | AGD_ADM.1 | Administrator guidance |
| 9 | AGD_USR.1 | User guidance |
| **Class ATE: Tests** | | |
| 10 | ATE_COV.1 | Evidence of coverage |
| 11 | ATE_FUN.1 | Functional testing |
| 12 | ATE_IND.2 | Independent testing- sample |
| **Class AVA: Vulnerability Assessment** | | |
| 13 | AVA_SOF.1 | Strength of TOE security function evaluation |
| 14 | AVA_VLA.1 | Developer vulnerability analysis |

### 5.2.1 Configuration management (ACM)

#### Configuration Items (ACM_CAP.2)

| | |
|---|---|
| **ACM_CAP.2.1D** | The developer shall provide a reference for the TOE. |
| **ACM_CAP.2.2D** | The developer shall use a CM system. |
| **ACM_CAP.2.3D** | The developer shall provide CM documentation. |
| **ACM_CAP.2.1C** | The reference for the TOE shall be unique to each version of the TOE. |
| **ACM_CAP.2.2C** | The TOE shall be labelled with its reference. |
| **ACM_CAP.2.3C** | The CM documentation shall include a configuration list. |
| **ACM_CAP.2.4C** | The configuration list shall describe the configuration items that comprise the TOE. |
| **ACM_CAP.2.5C** | The CM documentation shall describe the method used to uniquely identify the configuration items. |
| **ACM_CAP.2.6C** | The CM system shall uniquely identify all configuration items. |

### 5.2.2 Delivery and operation (ADO)

#### Delivery procedures (ADO_DEL.1)

| | |
|---|---|
| **ADO_DEL.1.1D** | The developer shall document procedures for delivery of the TOE or parts of it to the user. |
| **ADO_DEL.1.2D** | The developer shall use the delivery procedures. |
| **ADO_DEL.1.1C** | The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site. |

#### Installation, generation, and start-up procedures (ADO_IGS.1)

| | |
|---|---|
| **ADO_IGS.1.1D** | The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE. |
| **ADO_IGS.1.1C** | The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE. |

### 5.2.3 Development (ADV)

#### Informal functional specification (ADV_FSP.1)

| | |
|---|---|
| **ADV_FSP.1.1D** | The developer shall provide a functional specification. |
| **ADV_FSP.1.1C** | The functional specification shall describe the TSF and its external interfaces using an informal style. |
| **ADV_FSP.1.2C** | The functional specification shall be internally consistent. |
| **ADV_FSP.1.3C** | The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing details of effects, exceptions and error messages, as appropriate. |
| **ADV_FSP.1.4C** | The functional specification shall completely represent the TSF. |

#### Descriptive high-level design (ADV_HLD.1)

| | |
|---|---|
| **ADV_HLD.1.1D** | The developer shall provide the high-level design of the TSF. |
| **ADV_HLD.1.1C** | The presentation of the high-level design shall be informal. |
| **ADV_HLD.1.2C** | The high-level design shall be internally consistent. |
| **ADV_HLD.1.3C** | The high-level design shall describe the structure of the TSF in terms of subsystems. |
| **ADV_HLD.1.4C** | The high-level design shall describe the security functionality provided by each subsystem of the TSF. |
| **ADV_HLD.1.5C** | The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software. |
| **ADV_HLD.1.6C** | The high-level design shall identify all interfaces to the subsystems of the TSF. |
| **ADV_HLD.1.7C** | The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible. |

#### Informal correspondence demonstration (ADV_RCR.1)

| | |
|---|---|
| **ADV_RCR.1.1D** | The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided. |
| **ADV_RCR.1.1C** | For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation. |

#### Informal TOE security policy model (ADV_SPM.1)

| | |
|---|---|
| **ADV_SPM.1.1D** | The developer shall provide a TSP model. |
| **ADV_SPM.1.2D** | The developer shall demonstrate correspondence between the functional specification and the TSP model. |
| **ADV_SPM.1.1C** | The TSP model shall be informal. |
| **ADV_SPM.1.2C** | The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modelled. |
| **ADV_SPM.1.3C** | The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modelled. |
| **ADV_SPM.1.4C** | The demonstration of correspondence between the TSP model and the functional specification shall show that all of the security functions in the functional specification are consistent and complete with respect to the TSP model. |

### 5.2.4 Guidance documents (AGD)

#### Administrator guidance (AGD_ADM.1)

| | |
|---|---|
| **AGD_ADM.1.1D** | The developer shall provide administrator guidance addressed to system administrative personnel. |
| **AGD_ADM.1.1C** | The administrator guidance shall describe the administrative functions and interfaces available to the administrator of the TOE. |
| **AGD_ADM.1.2C** | The administrator guidance shall describe how to administer the TOE in a secure manner. |
| **AGD_ADM.1.3C** | The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment. |
| **AGD_ADM.1.4C** | The administrator guidance shall describe all assumptions regarding user behaviour that are relevant to secure operation of the TOE. |
| **AGD_ADM.1.5C** | The administrator guidance shall describe all security parameters under the control of the administrator, indicating secure values as appropriate. |
| **AGD_ADM.1.6C** | The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF. |
| **AGD_ADM.1.7C** | The administrator guidance shall be consistent with all other documentation supplied for evaluation. |
| **AGD_ADM.1.8C** | The administrator guidance shall describe all security requirements for the IT environment that are relevant to the administrator. |

#### User guidance (AGD_USR.1)

| | |
|---|---|
| **AGD_USR.1.1D** | The developer shall provide user guidance. |
| **AGD_USR.1.1C** | The user guidance shall describe the functions and interfaces available to the non-administrative users of the TOE. |
| **AGD_USR.1.2C** | The user guidance shall describe the use of user-accessible security functions provided by the TOE. |
| **AGD_USR.1.3C** | The user guidance shall contain warnings about user-accessible functions and privileges that should be controlled in a secure processing environment. |
| **AGD_USR.1.4C** | The user guidance shall clearly present all user responsibilities necessary for secure operation of the TOE, including those related to assumptions regarding user behaviour found in the statement of TOE security environment. |
| **AGD_USR.1.5C** | The user guidance shall be consistent with all other documentation supplied for evaluation. |
| **AGD_USR.1.6C** | The user guidance shall describe all security requirements for the IT environment that are relevant to the user. |

### 5.2.5 Tests (ATE)

#### Evidence of coverage (ATE_COV.1)

| | |
|---|---|
| **ATE_COV.1.1D** | The developer shall provide evidence of the test coverage. |
| **ATE_COV.1.1C** | The evidence of the test coverage shall show the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification. |

#### Functional testing (ATE_FUN.1)

| | |
|---|---|
| **ATE_FUN.1.1D** | The developer shall test the TSF and document the results. |
| **ATE_FUN.1.2D** | The developer shall provide test documentation. |
| **ATE_FUN.1.1C** | The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results. |
| **ATE_FUN.1.2C** | The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed. |
| **ATE_FUN.1.3C** | The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on the results of other tests. |
| **ATE_FUN.1.4C** | The expected test results shall show the anticipated outputs from a successful execution of the tests. |
| **ATE_FUN.1.5C** | The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified. |

#### Independent testing - sample (ATE_IND.2)

| | |
|---|---|
| **ATE_IND.2.1D** | The developer shall provide the TOE for testing. |
| **ATE_IND.2.1C** | The TOE shall be suitable for testing. |
| **ATE_IND.2.2C** | The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF. |

### 5.2.6 Vulnerability Analysis (AVA)

#### Strength of TOE security functions (AVA_SOF.1)

| | |
|---|---|
| **AVA_SOF.1.1D** | The developer shall perform a strength of TOE security function analysis for each mechanism identified in the ST as having a strength of TOE security function claim. |
| **AVA_SOF.1.1C** | For each mechanism with a strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the minimum strength level defined in the PP/ST. |
| **AVA_SOF.1.2C** | For each mechanism with a specific strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the specific strength of function metric defined in the PP/ST. |

#### Vulnerability analysis (AVA_VLA.1)

| | |
|---|---|
| **AVA_VLA.1.1D** | The developer shall perform and document an analysis of the TOE deliverables searching for obvious ways in which a user can violate the TSP. |
| **AVA_VLA.1.2D** | The developer shall document the disposition of obvious vulnerabilities. |
| **AVA_VLA.1.1C** | The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE. |

## 5.3 Security Requirements for the IT Environment

The TOE has no requirements for the IT environment.

## 5.4 Security Requirements for the Non-IT Environment

The TOE has the following security requirements for the Non-IT Environment.

**ENV_NONIT.1 Training of Destroy Users**

The TOE environment shall ensure that operators of the TOE are aware of, and if necessary trained, to use the TOE correctly, securely, and in accordance with defined policies.

**ENV_NONIT.2 Limiting Physical Access**

The TOE environment shall ensure that physical access to the media containing the TOE software is limited to only those people explicitly authorised access.

**ENV_NONIT.3 Verification of BIOS configuration**

The TOE environment shall ensure that operators of the TOE are aware of the need to ensure that the BIOS of the machine containing the hard drives to be sanitised has been correctly configured to know that all hard disks have been installed and procedures are in place to ensure that hard disk(s) with configured Host Protected Area(s) (HPA) have the HPA restored to the manufacturer's specification or the HPA is cleared.

# 6 TOE Summary Specification

This section presents the Security Functions implemented by the TOE and the Assurance Measures applied to ensure their correct implementation.

## 6.1 IT Security Functions

This section presents the security functions performed by the TOE and provides a mapping between the identified security functions and the Security Functional Requirements that it must satisfy.

**Table 10: IT Security Functions**

| IT Security Function Label | IT Security Function Description |
|---|---|
| **WRITE_K** | Write Known Data<br>The TOE uses low level write functions to write either 0x0 (ASCII Character 0) or 0xFF (ASCII Character 255) to each hard disk. The parameters for the write function are determined by the Destroy BIOS Query function. A failure of the write function can be detected and the function repeated based on both an individual and cumulative error count. |
| **WRITE_R** | Write Random Data<br>The TOE uses low level write functions writes pseudo-random ACSII characters to the each hard disk. The random data is generated from a pseudo-random number returned by a call to a standard C code library. The parameters for the write function are determined by the Destroy BIOS Query function. A failure of the write function can be detected and the function repeated based on both an individual and cumulative error count. |
| **VALID_K** | Validate Known Data<br>The TOE uses low level read functions to read data from each hard disk and validate that the data read matches the known data previously written by the Write Known Data function. The parameters for the write function are determined by the Destroy BIOS Query function. A failure of the read function can be detected and the function repeated based on both an individual and cumulative error count. |
| **SELF_INT** | Program Integrity Check<br>The TOE includes an integrity checking utility that calculates the MD5 hash of the TOE executable. The utility compares the calculated hash value with the stored hash value, termed the Destroy Validation Key. The utility also displays the value of the calculated hash on the screen to provide a mechanism for manual validation of the calculated hash against a printed copy of the Destroy Validation Key. |
| **REPORT** | Sanitisation Reporting<br>The TOE provides the Destroy Operator with on-going error reporting to the screen during the sanitisation process, and a final screen-based report of the overall outcome (success or failure) of the sanitisation process. The error count is also maintained to determine when a sector will be considered "bad" (after four attempts) and when the sanitisation will be considered to have "failed' (after three attempts to read or write a bad sector). |

| IT Security Function Label | IT Security Function Description |
|---|---|
| **BIOS_Q** | Destroy BIOS Query<br>On start-up of the TOE executable, the TOE interrogates the BIOS of the host PC to determine the reported number of drives, and for each hard drive:<br>• the reported capacity of the hard drive(s);<br>• the reported total number of heads;<br>• the reported total number of cylinders;<br>• the reported number of sectors.<br>The TOE then physically checks each hard disk using the Destroy BIOS Query Function, to determine the actual physical capacity, which may differ from the BIOS settings. The configuration of the hard drive(s) in the BIOS does not need to match the actual drive parameters. The only requirement is that the BIOS is configured to know that all hard disks are installed. The TOE then proceeds to overwrite the entire drive based on the actual physical capacity determined on start-up. This will include any hidden tracks (cylinders) found on each hard disk by Destroy. |

## 6.2 Assurance Measures

The TOE claims to satisfy the assurance requirements for the Common Criteria Evaluation Assurance Level EAL2 augmented (CC EAL2+). The augmented component is ADV_SPM.1. This section identifies the Configuration Management, System Development Procedures, System Test Documentation and System Installation and Guidance Documentation measures applied by TOE to satisfy the CC EAL2+ assurance requirements defined in the CC Part3 [CC3].

**Table 11: Assurance Measures**

| Assurance Measure Label | Assurance Measure Description |
|---|---|
| **CM_DOC** | Configuration management documentation that includes a configuration list, a description of the configuration items comprising the TOE and a description of the method used to uniquely identify the configuration items.<br><br>Document(s) title:<br>Production Guide for Destroy Version 2.01 & Destroy Lite Version 2.01 |
| **DEL_DOC** | Delivery documentation that describes all procedures necessary to maintain security for distribution of the TOE to a user's site.<br><br>Document(s) title:<br>Distribution Policy for Destroy Version 2.01 & Destroy Lite Version 2.01 |
| **IGS_DOC** | Installation and generation documentation that describes the steps necessary for secure installation, generation and startup of the TOE.<br><br>Document(s) title:<br>Production Guide for Destroy Version 2.01 & Destroy Lite Version 2.01<br>Operations Guide for Destroy Version 2.01 & Destroy Lite Version 2.01<br>Operations Guide for Destroy Version 2.01<br>Operations Guide for Destroy Lite Version 2.01 |
| **FUN_SPEC** | Functional specification that describes the TSF and its external interfaces and the purpose and method of use of external TSF interfaces, including details of effects, exceptions and error messages.<br><br>Document(s) title:<br>Functional Specification for Destroy Version 2.01 & Destroy Lite Version 2.01 |
| **HLD_DOC** | High-level design that describes the structure of the TSF in terms of sub-systems and describes the security functionality provided by each sub-system.<br><br>Document(s) title:<br>High Level Design for Destroy Version 2.01 & Destroy Lite Version 2.01 |
| **RCR_DOC** | Representation correspondence analysis that, for each adjacent pair TSF representations, demonstrates that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.<br><br>Document(s) title:<br>Representation Correspondence for Destroy Version 2.01 & Destroy Lite Version 2.01 |
| **SEC_POL** | Informal security policy model that describes the rules and characteristics of all policies of the TSP that can be modelled and demonstrates the correspondence between the TSP model and the functional specification.<br><br>Document(s) title:<br>Security Policy Model for Destroy Version 2.01 & Destroy Lite Version 2.01 |

| Assurance Measure Label | Assurance Measure Description |
|---|---|
| **ADMIN** | Administrator guidance that describes the administrative functions and interfaces available to the administrator of the TOE, describes how to administer the TOE in a secure manner, describes warnings about functions and privileges that should be controlled in a secure processing environment, describes all assumptions about user behaviour relevant to secure operation, describes all security parameters under the control of the administrator, and describes each type of security relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.<br><br>Document(s) title:<br>Operations Guide for Destroy Version 2.01 & Destroy Lite Version 2.01<br>Operations Guide for Destroy Version 2.01<br>Operations Guide for Destroy Lite Version 2.01 |
| **USER** | User guidance that describes the functions and interfaces available to the non-administrative users of the TOE, describes the use of user accessible security functions, describes warnings about user accessible functions and privileges that should be controlled in a secure processing environment, and describes all user responsibilities necessary for secure operation of the TOE.<br><br>Document(s) title:<br>Operations Guide for Destroy Version 2.01 & Destroy Lite Version 2.01<br>Operations Guide for Destroy Version 2.01<br>Operations Guide for Destroy Lite Version 2.01 |
| **TEST_COV** | Test evidence that shows the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification.<br><br>Document(s) title:<br>Test Coverage for Destroy Version 2.01 & Destroy Lite Version 2.01 |
| **TEST_DOC** | Test documentation consisting of test plans, test procedure descriptions, expected test results and actual test results. The test plan identifies the security functions to be tested and the goal of the tests to be performed. The test procedure descriptions identify the tests to be performed and describe the scenarios for testing each security function. The expected test results show the anticipated outputs from successful test execution. The actual test results demonstrate that each tested security function behaved as specified.<br><br>Document(s) title:<br>Testing for Destroy Version 2.01 & Destroy Lite Version 2.01<br>Testing for Destroy Version 2.01 & Destroy Lite Version 2.01 Expected Results<br>Testing for Destroy Version 2.01 & Destroy Lite Version 2.01 Actual Results<br>Testing for Publishing a Destroy Version 2.01 Disk & Destroy Lite Version 2.01 Disk<br>Testing for Destroy Validation Check Version (DVC) 2.01 & Destroy Lite Validation Check (DVC) Version 2.01 |
| **TEST_DOC** | The TOE and necessary supporting infrastructure suitable for testing.<br><br>Document(s) title & Equipment:<br>Test Plan for Destroy Version 2.01 & Destroy Lite Version 2.01 |
| **VLA_DOC** | A vulnerability analysis that shows that for all identified vulnerabilities, the vulnerability cannot be exploited in the intended environment of the TOE.<br><br>For each mechanism identified in the Security Target, an analysis shows that the claimed strength of TOE security function meets or exceeds the minimum strength level defined in the Security Target.<br><br>Document(s) title:<br>Vulnerability Analysis for Destroy Version 2.01 & Destroy Lite Version 2.01 |

# 7 PP Claims

This Security Target was not written to address a published Protection Profile.

# 8 Rationale

## 8.1 Security Objectives Rationale

The purpose of this rationale is to demonstrate that the identified security objectives are *suitable*, that is they are *sufficient* to address the security needs, and that they are *necessary*, i.e. there are no redundant security objectives.

### 8.1.1 All Assumptions, Threats and Policies Addressed

The need to demonstrate that there are no redundant security objectives is satisfied as follows:

- The first section (Table 12) shows that all of the secure usage assumptions, threats to security, and organisational security policies have been addressed.
- The second section shows that each security objective counters at least one assumption, policy, or threat.

**Table 12: Mapping of Assumptions, Threats, and OSPs to Security Objectives**

| Threat Label | Associated Security Objective |
|---|---|
| A.ADMIN_DOCS | OE.OPERATE |
| A.NO_EVIL | OE.OPERATE |
| A.ATTACK | O.SANITISE<br>O.INTEGRITY<br>OE.OPERATE<br>OE.MEDIA_ACCESS |
| A.NO_SOFTWARE | OE.OPERATE |
| A.NO_NETWORK | OE.OPERATE |
| A.BOOTABLE | OE.OPERATE<br>OE.MEDIA_ACCESS |
| A.BIOS_CONFIG | OE.OPERATE<br>OE.BIOS |
| T.DATA_ACCESS | O.SANITISE |
| T.DATA_DELETED | O.SANITISE |
| T.DATA_FORMAT | O.SANITISE |
| T.DATA_HIDDEN | O.SANITISE<br>OE.BIOS |
| T.MODIFY | O.INTEGRITY<br>OE.MEDIA_ACCESS |
| T.UNAWARE | O.NOTIFY |
| P.DISPOSAL | O.SANITISE<br>OE.OPERATE |
| P.INTEGRITY_CHECK | O.INTEGRITY<br>OE.OPERATE |

Table 13 shows that there are no unnecessary IT security objectives.

**Table 13: Mapping of Security Objectives to Threats, Policies and Assumptions**

| Objective Label | Threat / Policy/ Assumption |
|---|---|
| O.SANITISE | A.ATTACK<br>T.DATA_ACCESS<br>T.DATA_DELETED<br>T.DATA_FORMAT<br>T.DATA_HIDDEN<br>P.DISPOSAL |
| O.INTEGRITY | A.ATTACK<br>T.MODIFY<br>P.INTEGRITY_CHECK |
| O.NOTIFY | T.UNAWARE |
| OE.OPERATE | A.ADMIN_DOCS<br>A.NO_EVIL<br>A.ATTACK<br>A.NO_SOFTWARE<br>A.NO_NETWORK<br>A.BOOTABLE<br>A.BIOS_CONFIG<br>P.DISPOSAL<br>P.INTEGRITY_CHECK |
| OE.MEDIA_ACCESS | A.ATTACK<br>A.BOOTABLE<br>T.MODIFY |
| OE.BIOS | A.BIOS_CONFIG<br>T.DATA_HIDDEN |

### 8.1.2 Security Objectives are Sufficient

The following arguments are provided in Table 14 to demonstrate the sufficiency of the Security Objectives outlined above.

**Table 14: Sufficiency of Security Objectives**

| Threat Label | Argument to support Security Objective sufficiency |
|---|---|
| A.ADMIN_DOCS | This assumption is upheld by the objective OE.OPERATE as Destroy Users are required to ensure that the TOE is installed, managed and operated in accordance with TOE documentation. |
| A.NO_EVIL | This assumption is upheld by the objective OE.OPERATE as those responsible for the TOE are required to ensure that the TOE is installed, managed and operated in accordance with TOE documentation and in a manner consistent with organisational policies. In addition, Destroy Operators will have undergone appropriate background checks, psychological assessments and security clearances, as required. |
| A.ATTACK | This assumption is upheld by the following security objectives:<br>• O.SANITISE which ensures that attackers cannot gain access to data stored on hard drives that are to be redeployed, transferred out of the organisation's control, or discarded;<br>• O.INTEGRITY which ensures that attackers cannot compromise the integrity of the Destroy software;<br>• OE.OPERATE which ensures that the TOE is installed, managed and operated in a manner consistent with the documentation; and<br>• OE.MEDIA_ACCESS which ensures that an attacker cannot gain physical access to the media containing the TOE. |
| A.NO_SOFTWARE | This assumption is upheld by the objective OE.OPERATE as those responsible for the TOE are required to ensure that the TOE is installed, managed and operated in accordance with TOE documentation and in a manner consistent with organisational policies, which requires that no other software is operational while the TOE is being used. |
| A.NO_NETWORK | This assumption is upheld by the objective OE.OPERATE as those responsible for the TOE are required to ensure that the TOE is installed, managed and operated in accordance with TOE documentation and in a manner consistent with organisational policies, which requires that there is no network connectivity while the TOE is being used. |
| A.BOOTABLE | This assumption is upheld by the following security objectives:<br>• OE.OPERATE which ensures that the TOE is installed, managed and operated in a manner consistent with the documentation; and<br>• OE.MEDIA_ACCESS which ensures that physical access to the TOE media is controlled in accordance with defined policies. |
| A.BIOS_CONFIG | This assumption is upheld by the following security objectives:<br>• OE.OPERATE which ensures that the TOE is installed, managed and operated in a manner consistent with the documentation; and<br>• OE.BIOS which ensures that the BIOS of the machine containing the hard drive(s) to be sanitised has been correctly configured to know that all hard disks have been installed and the Host Protected Area, if configured, has been restored to the manufacturer's specification or removed. |

| Threat Label | Argument to support Security Objective sufficiency |
|---|---|
| T.DATA_ACCESS | The threat of an unauthorised person gaining access to sensitive data on a hard drive that is to be redeployed, transferred out of the organisation's control, or discarded is countered by the objective O.SANITISE, which ensures that the data is not available for access due to sanitisation. |
| T.DATA_DELETED | The threat of an unauthorised person attempting to recover data remaining after the data has been deleted from a hard drive that is to be redeployed, transferred out of the organisation's control, or discarded is countered by the objective O.SANITISE, which ensures that the data is not available for access due to sanitisation. |
| T.DATA_FORMAT | The threat of an unauthorised person attempting to recover data remaining after formatting of a hard drive that is to be redeployed, transferred out of the organisation's control, or discarded is countered by the objective O.SANITISE, which ensures that the data is not available for access due to sanitisation. |
| T.DATA_HIDDEN | The threat of an unauthorised person attempting to recover data from a hard drive that is to be redeployed, transferred out of the organisation's control, or discarded is countered by the following objectives:<br>• O.SANITISE, which ensures that the data is not available for access due to sanitisation; and<br>• OE.BIOS, which ensures that the BIOS has been correctly configured to know that all hard disks have been installed and the Host Protected Area, if configured, has been restored to the manufacturer's specification or removed. |
| T.MODIFY | The threat of an unauthorised person attempting to modify the software comprising the TOE is countered by the objective O.INTEGRITY, which ensures that any loss of integrity affecting TOE operation can be detected.<br>In addition, the threat of an unauthorised person attempting to modify the integrity checking software is countered by the objective OE.MEDIA_ACCESS which requires that media containing the product executables is appropriately protected. |
| T.UNAWARE | The threat of an authorised user of the TOE being unaware of a failure to completely sanitise a hard drive is countered by the objective O.NOTIFY, which ensures that a user is notified of the success and/or failure of the operation of the TOE. |
| P.DISPOSAL | The OSP requirement for appropriate management of the identification, disposal and sanitisation of hard drives is met by the following objectives:<br>• O.SANITISE, which ensures that the functionality to sanitise hard drives is provided; and<br>• OE.OPERATE, which ensures that TOE users have appropriate guidance on the use of the TOE. |
| P.INTEGRITY_CHECK | The OSP requirement for checking the integrity of the TOE is met by the following objectives:<br>• O.INTEGRITY, which ensures that the functionality to detect modifications of the TOE is provided; and<br>• OE.OPERATE, which ensures that TOE users have appropriate guidance on the use of the TOE. |

## 8.2 Security Requirements Rationale

### 8.2.1 Suitability of the Security Requirements

The purpose of this section is to show that the identified security requirements are *suitable* to meet the security objectives. Table 15 and Table 16 show that each security requirement is *necessary,* that is, each security objective is addressed by at least one security requirement and vice versa. Note that some objectives are partially satisfied by the TOE and partially satisfied by the IT environment. Security Objectives for the TOE are satisfied by Common Criteria functional components. Security Objectives for the Environment are satisfied by IT requirements for the environment.

**Table 15: Mapping of Security Objectives to Security Requirements**

| Objectives | Requirements |
|---|---|
| O.SANITISE | FDP_RIP.2<br>FPT_RCV.4 |
| O.INTEGRITY | FCS_COP.1<br>FPT_PHP.1<br>FPT_AMT.1<br>FPT_TST.1 |
| O.NOTIFY | FAU_ARP.1<br>FAU_GEN.1<br>FAU_SAA.1<br>FPT_RCV.4 |
| OE.OPERATE | ENV_NONIT.1 |
| OE.MEDIA_ACCESS | ENV_NONIT.2 |
| OE.BIOS | ENV_NONIT.3 |

**Table 16: Mapping of Security Requirements to Security Objectives**

| Requirements | Objective |
|---|---|
| FAU_ARP.1 | O.NOTIFY |
| FAU_GEN.1 | O.NOTIFY |
| FAU_SAA.1 | O.NOTIFY |
| FCS_COP.1 | O.INTEGRITY |
| FDP_RIP.2 | O.SANITISE |
| FPT_AMT.1 | O.INTEGRITY |
| FPT_PHP.1 | O.INTEGRITY |
| FPT_RCV.4 | O.SANITISE<br>O.NOTIFY |
| FPT_TST.1 | O.INTEGRITY |
| ENV_NONIT.1 | OE.OPERATE |
| ENV_NONIT.2 | OE.MEDIA_ACCESS |
| ENV_NONIT.3 | OE.BIOS |

### 8.2.2 Sufficiency of the Security Requirements

The following table shows that security requirements are *sufficient* to satisfy the TOE security objectives, whether in a principal or supporting role.

**Table 17: Sufficiency of Security Requirements**

| Objectives | Argument to support sufficiency of Security Requirements |
|---|---|
| O.SANITISE | The objective to prevent unauthorised access to data stored on a hard drive that has been redeployed, transferred out of the organisation's control, or discarded is met by the following security requirements:<br>• FDP_RIP.2, which ensures that once a resource (defined here as a hard drive) has been de-allocated, any previous information content is made unavailable; and<br>• FDP_RCV.4, which ensures that the failure of any aspect of the sanitisation process will result in recovery to a secure state (either completing the sanitisation or informing the user of the failure). |
| O.INTEGRITY | The objective to provide a means of detecting loss of integrity of the TOE that may affect operation is met by the following security requirements:<br>• FCS_COP.1, which requires the TOE to calculate and verify MD5 hash values to calculate and verify an MD5 hash of the Destroy program executable;<br>• FPT_PHP.1, which ensures that efforts to physically tamper with the TOE can be readily detected;<br>• FPT_AMT.1 as a dependency on FPT_TST.1 requires abstract machine testing; and<br>• FPT_TST.1, which ensures that the TOE provides the capability for authorised users to verify the integrity of the executable code of the TOE. |
| O.NOTIFY | The objective to provide a means of notifying authorised users of the success and/or failure to sanitise a hard drive is met by the following requirements:<br>• FAU_ARP.1, FAU_GEN.1, and FAU_SAA.1, which ensure that the TOE provides an authorised user with a screen-based report of the outcome of TOE operations, including success and/or failure, and can also report on the accumulation of particular failure events; and<br>• FPT_RCV.4, which ensures that any failure of the TOE functions will be reported to the authorised user to inform the user of the security state of the hard drive. |
| OE.OPERATE | This objective is met by the security requirement ENV_NONIT.1 as it requires that TOE users are appropriately knowledgeable and/or trained in the use of the TOE, its accompanying documentation and any relevant organisational policies. |
| OE.MEDIA_ACCESS | This objective is met by the security requirement ENV_NONIT.2 as it requires that appropriate physical protection mechanism exist to prevent unauthorised access to the media containing the TOE software. |
| OE.BIOS | This objective is met by the security requirement ENV_NONIT.3 as it requires that authorised users of the TOE ensure that the BIOS for the PC containing the hard drives to be sanitised is correctly configured to know that all hard disk have been installed and the Host Protected Area, if configured, has been restored to the manufacturer's specification or removed. |

### 8.2.3 Satisfaction of Dependencies

Table 18 shows the dependencies between the functional requirements. All of the dependencies are satisfied. Note that:

(*) indicates that this dependency is not satisfied by the TOE. Refer to the supporting rationale following Table 18.

**Table 18: Dependency Analysis**

| Component Reference | Requirement | Dependencies | Dependency Reference |
|---|---|---|---|
| **Functional Requirements** | | | |
| 1 | FAU_ARP.1 | FAU_SAA.1 | 3 |
| 2 | FAU_GEN.1 | FPT_STM.1 | * |
| 3 | FAU_SAA.1 | FAU_GEN.1 | 2 |
| 4 | FCS_COP.1 | FCS_CKM.1, FCS_CKM.4, FMT_MSA.2 | *,*,* |
| 5 | FDP_RIP.2 | None | - |
| 6 | FPT_AMT.1 | None | - |
| 7 | FPT_PHP.1 | FMT_MOF.1 | * |
| 8 | FPT_RCV.4 | ADV_SPM.1 | 16 |
| 9 | FPT_TST.1 | FPT_AMT.1 | 6 |
| **Assurance Requirements** | | | |
| 10 | ACM_CAP.2 | None | - |
| 11 | ADO_DEL.1 | None | - |
| 12 | ADO_IGS.1 | AGD_ADM.1 | 17 |
| 13 | ADV_FSP.1 | ADV_RCR.1 | 15 |
| 14 | ADV_HLD.1 | ADV_FSP.1, ADV_RCR.1 | 13, 15 |
| 15 | ADV_RCR.1 | None | - |
| 16 | ADV_SPM.1 | ADV_FSP.1 | 13 |
| 17 | AGD_ADM.1 | ADV_FSP.1 | 13 |
| 18 | AGD_USR.1 | ADV_FSP.1 | 13 |
| 19 | ATE_COV.1 | ADV_FSP.1, ATE_FUN.1 | 13, 20 |
| 20 | ATE_FUN.1 | None | - |
| 21 | ATE_IND.2 | ADV_FSP.1, AGD_ADM.1, AGD_USR.1, ATE_FUN.1 | 13, 17, 18, 20 |
| 22 | AVA_SOF.1 | ADV_FSP.1, ADV_HLD.1 | 13, 14 |
| 23 | AVA_VLA.1 | ADV_FSP.1, ADV_HLD.1, AGD_ADM.1, AGD_USR.1 | 13, 14, 17, 18 |

The following dependencies are not satisfied in this Security Target because they are not considered relevant to the TOE for the provided reasons:

- FPT_STM.1 Reliable time stamps

  The dependency on FPT_STM.1 comes from the audit requirement FAU_GEN.1 but for this TOE audit records are created and provided to the user in real-time and so a time stamp is not required.

- FMT_MOF.1 Management of security functions behaviour

  The dependency on FMT_MOF.1 comes from the self-protection requirement FPT_PHP.1. According to the CC Part 2, the reliance on FMT_MOF.1 is based on the requirement to manage the reporting of attempted attacks and the list of devices that can report attacks. For this TOE, reporting of attempted attacks is only provided to the TOE user and the TOE can only detect attacks on its own executable code. As none of these features are configurable, there is no need to include a requirement for the management of these functions.

- FCS_CKM.1 Cryptographic Key Generation

  The dependency on FCS_CKM.1 comes from the cryptographic operation requirement FCS_COP.1. As the instance of FCS_COP.1 has been assigned values associated with performing MD5 hashing operations, no key generation is required which means there is no requirement for the FCS_CKM.1 component.

- FCS_CKM.4 Cryptographic Key Destruction

  The dependency on FCS_CKM.4 comes from the cryptographic operation requirement FCS_COP.1. As the instance of FCS_COP.1 has been assigned values associated with performing MD5 hashing operations, no key destruction is required which means there is no requirement for the FCS_CKM.4 component.

- FMT_MSA.2 Secure security attributes

  FMT_MSA.2 is identified as a dependency for FCS_COP.1. The intent of FMT_MSA.2 is that values for security attributes must not violate the TSP. In the context of the FCS family, FMT_MSA.2 requires that the combination of cryptographic security attributes such as key length, key validity period and key use (e.g. digital signature, key encryption, data encryption) may only be set to values which maintain the 'secure state' of the TOE. As there are no configurable cryptographic security attributes, the requirement FCS_COP.1 is met without satisfying this dependency.

## 8.3 TOE Summary Specification Rationale

### 8.3.1 IT security functions satisfy the SFRs.

The following two tables show that each SFR is mapped to at least one IT security function and each IT security function is mapped to at least one SFR.

**Table 19: Mapping of SFRs to IT Security Functions**

| Security Functional Requirement | IT Security Function |
|---|---|
| FAU_ARP.1 | REPORT |
| FAU_GEN.1 | REPORT<br>BIOS_Q |
| FAU_SAA.1 | REPORT |
| FCS_COP.1 | SELF_INT |
| FDP_RIP.2 | WRITE_K<br>WRITE_R<br>VALID_K<br>BIOS_Q |
| FPT_AMT.1 | BIOS_Q<br>WRITE_K<br>WRITE_R<br>VALID_K |
| FPT_PHP.1 | SELF_INT |
| FPT_RCV.4 | WRITE_K<br>WRITE_R<br>VALID_K<br>REPORT |
| FPT_TST.1 | BIOS_Q<br>SELF_INT<br>VALID_K |

**Table 20: Mapping of IT Security Functions to SFRs**

| IT Security Function | Security Functional Requirement |
|---|---|
| WRITE_K | FDP_RIP.2<br>FPT_AMT.1<br>FPT_RCV.4 |
| WRITE_R | FDP_RIP.2<br>FPT.AMT.1<br>FPT_RCV.4 |
| VALID_K | FDP_RIP.2<br>FPT_AMT.1<br>FPT_RCV.4<br>FPT_TST.1 |
| SELF_INT | FCS_COP.1<br>FPT_PHP.1<br>FPT_TST.1 |
| REPORT | FAU_ARP.1<br>FAU_GEN.1<br>FAU_SAA.1<br>FPT_RCV.4 |
| BIOS_Q | FAU_GEN.1<br>FDP_RIP.2<br>FPT_AMT.1<br>FPT_TST.1 |

### 8.3.2 IT Security Function Suitability

Table 21 provides appropriate justification that the IT Security Functions are suitable to meet the TOE Security Functional Requirement and that when implemented, contributes to meeting that requirement.

**Table 21: Suitability of IT Security Functions**

| Security Functional Requirement | Argument for suitability of IT Security Functions |
|---|---|
| FAU_ARP.1 | This TOE SFR is satisfied by the IT Security Function REPORT as the function provides the TOE user with a screen-based report of any errors, both individual and cumulative, as they occur during the sanitisation process. |
| FAU_GEN.1 | This TOE SFR is satisfied by the following IT Security Functions:<br>• REPORT, as the function provides the TOE user with a report of the outcome of the audit process including the overall success and/or failure, the outcome of individual steps, and the details of the hard drive that was sanitised; and<br>• BIOS_Q, as the function provides the capability to determine the parameters of the hard drive for sanitisation through the Destroy BIOS Query function. |
| FAU_SAA.1 | This TOE SFR is satisfied by the IT Security Function REPORT as the function provides a report of the accumulation and combination of sanitisation errors as they occur during the sanitisation process. |
| FCS_COP.1 | This TOE SFR is satisfied by the IT Security Function SELF_INT as the function provides authorised users with the capability to verify the integrity of stored TOE executable code through the use of an MD5 hash. |
| FDP_RIP.2 | This TOE SFR is satisfied by the following IT Security Functions:<br>• WRITE_K, as the function provides the capability to write known data (0x0 and 0xFF) to the hard disk;<br>• WRITE_R, as the function provides the capability to write pseudo-random data to the hard disk;<br>• VALID_K, as the function provides the capability to validate the known data that has been written to the hard disk; and<br>• BIOS_Q, as the function provides the capability to determine the parameters for the read and write functions. |

| Security Functional Requirement | Argument for suitability of IT Security Functions |
|---|---|
| FPT_AMT.1 | This TOE SFR is satisfied by the following IT Security Functions:<br>• BIOS_Q, as the function provides the capability to detect the presence of one or more hard disks in the computer;<br>• WRITE_K, as the function provides the capability to detect the presence of one or more hard disks in the computer;<br>• WRITE_R, as the function provides the capability to detect the presence of one or more hard disks in the computer; and<br>• VALID_K, as the function provides the capability to detect the presence of one or more hard disks in the computer. |
| FPT_PHP.1 | This TOE SFR is satisfied by the IT Security Function SELF_INT as the function provides the capability to detect attempts to physically tamper with the TOE through integrity checking of the TOE executable code and the independent validation of the checking program itself. |
| FPT_RCV.4 | This TOE SFR is satisfied by the following IT Security Functions:<br>• WRITE_K, as the function provides the capability to detect the failure of an attempt to write known data to the hard drive and to repeat the function;<br>• WRITE_R, as the function provides the capability to detect the failure of an attempt to write random data to the hard drive and to repeat the function;<br>• VALID_K, as the function provides the capability to detect either the failure to read data or the failure to read the correct data from the hard drive and to repeat the function; and<br>• REPORT, as the function provides the capability to count and manage the individual and cumulative errors, report all individual and cumulative failures and to unambiguously notify the TOE user of the outcome of the sanitisation operation. |
| FPT_TST.1 | This TOE SFR is satisfied by the following IT Security Functions:<br>• SELF_INT, as the function provides authorised users with the capability to verify the integrity of stored TOE executable code.<br>• BIOS_Q, as this function queries the BIOS to check the number and size of hard disks to be wiped. This function is automatically executed during startup of the TOE by an authorised user.<br>• VALID_K as the function reads back the data after the execution of the WRITE_K function to demonstrate the correct operation of the TSF to sanitise the hard disk(s). |

### 8.3.3 Demonstration of Mutual Support

The primary function of the TOE, namely to securely sanitise hard drives, is provided by an SFR from the FDP class. The SFRs selected from the FPT class provide integrity detection and secure recovery functions to ensure that the FDP function continues to operate in a secure manner. The SFR selected from the FCS class supports the integrity detection SFR from the FPT class. The SFRs selected from the FAU class provide notification functions in conjunction with the FDP and FPT components to ensure that the outcome of the sanitisation function, either success or failure, is unambiguously provided to the TOE user.

The dependency analysis provided at Table 18 and the analyses provided in Table 19,

Table 20 and Table 21 demonstrate that the IT security functions work together to satisfy the TSFs, that is, they demonstrate mutual support between function components.

By definition, all assurance requirements support all SFRs since they provide confidence in the correct implementation and operation of the SFRs.

This analysis of the security functional and assurance requirements demonstrates that there are no conflicts between requirements. Therefore, the security requirements together form a mutually supportive and consistent whole.

### 8.3.4 Assurance Security Requirements Rationale

Table 22 below shows that all Security Assurance Requirements are met by the assurance measures.

**Table 22: Mapping of SARs to Assurance Measures**

| Security Assurance Requirements | Assurance Measures |
|---|---|
| ACM_CAP.2 | CM_DOC |
| ADO_DEL.1 | DEL_DOC |
| ADO_IGS.1 | IGS_DOC |
| ADV_FSP.1 | FUN_SPEC |
| ADV_HLD.1 | HLD_DOC |
| ADV_RCR.1 | RCR_DOC |
| ADV_SPM.1 | SEC_POL |
| AGD_ADM.1 | ADMIN |
| AGD_USR.1 | USER |
| ATE_COV.1 | TEST_COV |
| ATE_FUN.1 | TEST_DOC |
| ATE_IND.2 | TEST_DOC |
| AVA_SOF.1 | VLA_DOC |
| AVA_VLA.1 | VLA_DOC |

Given that all Security Assurance Requirements are met by at least one assurance measure, and that the implementation of each assurance measure will be the subject of evaluation activities, it is concluded that all of the assurance measures will meet all of the Security Assurance Requirements.

Given the threats and security objectives outlined, it could be argued that the assurance level would be determined by the value of the assets the TOE is meant to protect. However, considering the value of the assets alone is not sufficient for determining the appropriate assurance level for the TOE. There are measures defined for the environment (defined in Section 3 and as requirements for the non-IT environment in Section 5.3) that significantly decrease the risks to the IT assets. These measures include:

1. It is intended that the TOE be executed in an environment that does not contain any other active software or network connectivity;
2. It is intended that the media containing the TOE software is physically protected and controlled;
3. There are no Strength of Function requirements.

Given that the residual risks to the IT assets have been partially (and significantly) mitigated by the security measures in the environment, the attractiveness of the assets can be considered to be similarly reduced. Thus, the combination of the reduced attractiveness of the IT assets and the security measures provided by the environment, it is considered that an EAL-2+ level of assurance is entirely appropriate for the intended application of the TOE.

The augmentation of CC EAL-2 comprised the addition of the component ADV_SPM.1. This component, an informal security policy model, was included to satisfy a dependency from the component FPT_RCV.4 which requires the definition of secure states.

### 8.3.5 Strength of function claims

At CC EAL-2+, the TOE security assurance requirements include the AVA_SOF.1 component. The minimum strength of function claim for the TOE security functional requirements is *SOF_basic*. This claim is appropriate due to the assumptions made in Section 3.1 regarding the motivation, expertise and resources. It is assumed that an attacker has a medium level of expertise and resources and a medium level of motivation. The TOE environment also provides for physical and logical protection of the TOE.

The WRITE_R function writes random numbers to the disk, generated by a pseudo-random number generator, and is thus a permutational mechanism. Therefore, consistent with the minimum strength of function claim for the TOE, the claim of SOF-basic is applicable to this mechanism.

Normally, an explicit strength of function claim would be appropriate for the TOE IT Security Function:

- SELF_INT.

SELF_INT implements the security functional requirements FPT_PHP.1 and FPT_TST.1, which are implemented through an MD5 hashing mechanism. As the MD5 hashing mechanism is cryptographic, an explicit strength of function claim is not appropriate for the SELF_INT function.

## 8.4 Rationale for Extensions

Not applicable.

## 8.5 PP Claims Rationale

This ST makes no PP conformance claim therefore no rationale is required.

# Appendix A - Acronyms

| | |
|---|---|
| CC | Common Criteria |
| EAL | Evaluation Assurance Level |
| IT | Information Technology |
| PP | Protection Profile |
| SF | Security Function |
| SFP | Security Function Policy |
| SOF | Strength of Function |
| ST | Security Target |
| TOE | Target of Evaluation |
| TSC | TSF Scope of Control |
| TSF | TOE Security Functions |
| TSFI | TSF Interface |
| TSP | TOE Security Policy |