

**Independent Validation & Verification (IV&V)**

**Of**

**Storage Media Archival Recovery Toolkit (SMART)**

**For Linux**

**Thomas Rude, CISSP  
Security Consultant**

**October 31, 2002**

# Contents

---

<b>1. Overview</b>	<b>3</b>
IV&V Defined	3
SMART for Linux	3
Verification Goals	3
<b>2. Test Bed</b>	<b>4</b>
Hardware	4
Software	4
Configuration	4
<b>3. Methodology</b>	<b>5</b>
Validation Methodology	5
<b>4. Results</b>	<b>6-7</b>
Summary	6
Interpretations	7
<b>5. Appendix A</b>	<b>8</b>
Step-by-Step	8

# Chapter 1 – Overview

---

## IV&V Defined

An Independent Validation and Verification is just that – a third party independent from the developer performs the validation and verification process for a target. Specifically in this case, an IV&V is a controlled process based on sound methodology that attempts to evaluate the accuracy of a data forensic software program. The validation component focuses on whether or not there is a demonstrated need for the program while the verification component focuses on testing the program to insure it works as intended and expected. There are many benefits of an IV&V, including;

- confidence in the program
- reduced cost
- adherence to standards

The actual testing of the program and publishing of findings can reduce the cost of the program to any organization if the testing methodology is sound and the findings show the program works as intended. An organization will not have to take the time (spend the resources), to test the program if a test has already been conducted – provided the test results can be duplicated. Further, a successful IV&V wherein the findings support accuracy may promote confidence in the program without having to rigorously test it before purchasing it.

The purpose of this document is to report the findings for the Independent Validation and Verification (IV&V) of *Storage Media Archival Recovery Toolkit (SMART)* for Linux by ASR Data Acquisition and Analysis, LLC. A quick market study reveals the validity of SMART and other similar data forensic programs; there is a clear and current need for these programs.

## SMART for Linux

SMART is a data forensic program that runs on both BeOS and Linux operating systems. It is developed by ASR Data Acquisition and Analysis, LLC. It is sold and marketed as a feature rich data forensic utility that has at a very minimum the following functionality;

- media hashing
- media imaging
- media restoring
- media wiping

## Verification Goals

This IV&V has the following goals;

- verify SMART hashing functionality
- verify SMART imaging functionality
- verify SMART restoring functionality
- verify SMART wiping functionality

For the purposes of this IV&V ‘verify’ means the functionality works as intended and advertised, I.E., ‘it works’.

## Chapter 2 – Test Bed

---

### Hardware

The following hardware was used;

- Generic PC
- Award Modular BIOS v6.00PG
- Primary Drive 0: Maxtor 6L060J3 60GB
- Primary Drive 1: Maxtor 6L040J2 40GB
- Secondary Drive 0: ATAPI Device (CD-ROM)
- Secondary Drive 1: Maxtor 6L020L1 20GB

### Software

The following software was used;

- Red Hat Linux 7.2
- BeOS Professional 5.03
- SMART October 17, 2002 Build
- Verification Tools;
  - md5sum (textutils package) v2.0, BeOS
  - dd (fileutils package) v4.0, BeOS
  - md5sum (textutils package) v2.0.14, Linux
  - dd (fileutils package) v4.1, Linux

### Configuration

The system was configured as follows;

- Primary Drive 0:
  - Lab Drive
  - 60GB capacity
  - Dual boot BeOS and Linux operating systems
- Primary Drive 1:
  - Evidentiary Drive
  - 40GB capacity
  - Linux operating system

Primary Drive 0 contained the lab drive used for testing SMART. This was the larger of the two drives and contained a free space area roughly 50GB in size. This area was used for receiving Evidentiary Drive data – images, hash values, etc. Primary Drive 0 was a dual boot system capable of booting either the BeOS or Red Hat Linux operating systems.

Primary Drive 1 was the Evidentiary Drive. It was a bootable system running Red Hat Linux 7.2.

## Chapter 3 – Methodology

---

### Verification Methodology

Verifying the four basic functionalities of SMART required a simple, yet sound, methodology. One goal was always in mind, and that was the methodology must be designed such that anyone could replicate it and achieve similar results.

- 1) Authenticate (hash) Primary Drive 1 (Evidentiary Drive)
  - 'md5sum' in BeOS
  - 'md5sum' in Red Hat Linux
  - SMART authenticate function
- 2) Acquire (image) Primary Drive 1 (Evidentiary Drive)
  - SMART acquire function
- 3) Authenticate (hash) SMART Created Image File Created in #2
  - 'md5sum' in BeOS
  - 'md5sum' in Red Hat Linux
  - SMART authenticate function
- 4) Wipe Primary Drive 1 (Evidentiary Drive)
  - SMART wipe function
- 5) Authenticate (hash) SMART Wiped Primary Drive 1 (Evidentiary Drive)
  - 'md5sum' in BeOS
  - 'md5sum' in Red Hat Linux
  - SMART authenticate function
- 6) Restore SMART Created Image File to Primary Drive 1 (Evidentiary Drive)
  - SMART restore function
- 7) Authenticate SMART Restored Primary Drive 1 (Evidentiary Drive)
  - 'md5sum' in BeOS
  - 'md5sum' in Red Hat Linux
  - SMART authenticate function

## Chapter 4 – IV&V Results

---

### IV&V Summary

The results of the IV&V are summarized in *Table 1* below;

	Primary Drive 1 (Evidentiary)	SMART Primary Drive 1 Image	SMART Primary Drive 1 Wiped	SMART Primary Drive 1 Restored
BeOS md5sum	f5f76c206e409625432e34f686295473	f5f76c206e409625432e34f686295473	783874834ab49ff4655e02cb69a6e295	f5f76c206e409625432e34f686295473
Linux md5sum	f5f76c206e409625432e34f686295473	f5f76c206e409625432e34f686295473	783874834ab49ff4655e02cb69a6e295	f5f76c206e409625432e34f686295473
SMART md5sum	f5f76c206e409625432e34f686295473	f5f76c206e409625432e34f686295473	783874834ab49ff4655e02cb69a6e295	f5f76c206e409625432e34f686295473

**Table 1** Summary of Authentication Values

Before the verification process was started these were the logical assumptions;

- All three hash values for Primary Drive 1 (Evidentiary) should be equal
- All three hash values for the SMART Created Image File should be equal
- All three hash values for the SMART Wiped Primary Drive 1(Evidentiary) should be equal
- All three hash values for the SMART Restored Primary Drive 1 (Evidentiary) should be equal

As shown in *Table 1* all hash values were equivalent where we would expect them to be equal. The 'md5sum' utility in both BeOS and Red Hat Linux and the authentication function in SMART each calculated the same value for the original Primary Drive 1 (Evidentiary Drive). The 'md5sum' utility in both BeOS and Red Hat Linux and the authentication function in SMART each calculated the same value for the Image File of the Primary Drive 1 (Evidentiary Drive) created by SMART. The 'md5sum' utility in both BeOS and Red Hat Linux and the authentication function in SMART each calculated the same value for the SMART wiped Primary Drive 1 (Evidentiary Drive). The 'md5sum' utility in both BeOS and Red Hat Linux and the authentication function in SMART each calculated the same value for the SMART restored Primary Drive 1 (Evidentiary Drive).

Note that there was only one utility used to create an image of Primary Drive 1 (Evidentiary Drive), and that utility was SMART. The reason for this was simple – if the hash value for the original Primary Drive 1 (Evidentiary Drive) was equal to the hash value calculated for the Image File created by SMART then a full image of Primary Drive 1 (Evidentiary Drive) was created by SMART. If the hash values were not equal to one another then there was a problem with the imaging process (acquisition function).

The 'dd' utility included in both BeOS and Red Hat Linux could also be used to create an image of Primary Drive 1 (Evidentiary Drive) if desired. This was not done here because it would have been redundant.

## Chapter 4 – IV&V Results

---

### Interpretations

From the hash value results shown in *Table 1* each of the four tested functionalities of SMART have been verified;

- Authentication Function = The hash value for the original Primary Drive 1 (Evidentiary) calculated by all three hashing programs ('md5sum' for both BeOS and Linux and SMART hash function) is the same across the board. Therefore, the conclusion is that the authentication function of SMART works as intended and is accurate.
- Image Function = The hash value for the Image File of Primary Drive 1 (Evidentiary) created by SMART calculated by all three hashing programs is equal to the hash value of the original Primary Drive 1 (Evidentiary) calculated by all three hashing programs. Therefore, the conclusion is that the image function of SMART works as intended and is accurate.
- Restore Function = The hash value for the SMART Restored Primary Drive 1 (Evidentiary) calculated by all three hashing programs is equal to one another and equal to the value of the original Primary Drive 1 (Evidentiary) calculated by all three hashing programs. Therefore, the conclusion is that the restore function of SMART works as intended and is accurate.
- Wipe Function = The hash value for the SMART Wiped Primary Drive 1 (Evidentiary) calculated by all three hashing programs is equal to one another. Further, this value is not equal to the original Primary Drive 1 (Evidentiary) hash value. Therefore, the conclusion is that the wipe function of SMART works as intended and is accurate.

SMART Function	Pass	Fail
Authenticate	X	
Image	X	
Restore	X	
Wipe	X	

**Table 2** *Verification Conclusion Summary*

Each of the four tested functions of SMART passed the verification process. Each function was found to be accurate.

## Chapter 5 – Appendix A

---

### Step-by-Step

- Power on PC
- Boot into BeOS
- Use 'md5sum' utility to hash physical device, Primary Drive 1 (Evidentiary)
- Reboot PC into Linux
- Use 'md5sum' utility to hash physical device, Primary Drive 1 (Evidentiary)
- Start SMART program
- In SMART right-click and select 'Produce Hash' on Primary Drive 1 (Evidentiary)
- Compare all three hash values, proceed if equivalent
- In SMART right-click and select 'Acquire' on Primary Drive 1 (Evidentiary)
- In SMART select 'Authenticate' 'Against a Disk' Primary Drive 1 (Evidentiary)
- Use 'md5sum' utility in Linux to hash SMART created Image File
- Reboot into BeOS
- Use 'md5sum' utility in BeOS to hash SMART created Image File
- Compare all three hash values
- Reboot into Linux
- Start SMART program
- In SMART right-click and select 'Wipe' on Primary Drive 1 (Evidentiary)
- In SMART right-click and select 'Produce Hash' on Primary Drive 1 (Evidentiary)
- Use 'md5sum' utility in Linux to hash SMART wiped Primary Drive 1 (Evidentiary)
- Reboot into BeOS
- Use 'md5sum' utility in BeOS to hash SMART wiped Primary Drive 1 (Evidentiary)
- Compare all three hash values
- Reboot into Linux
- Start SMART program
- Open IV&V Case, right-click on Image File, select 'Copy', 'Restore to Storage Devices' and select Primary Drive 1 (Evidentiary)
- In SMART right-click and select 'Produce Hash' on SMART Restored Primary Drive 1 (Evidentiary)
- Use 'md5sum' utility in Linux to hash SMART restored Primary Drive 1 (Evidentiary)
- Reboot into BeOS
- Use 'md5sum' utility in BeOS to hash SMART restored Primary Drive 1 (Evidentiary)