University of Glamorgan

Prifysgol Morgannwg

School of Computing

M.Sc. Project

A prototype of a client/server forensic program using XML

(Improving the performance
of the next generation
of digital forensic tools)

Nicholas Pringle

03118460

First Supervisor: *Dr Iain Sutherland*

Second Supervisor: *Duncan McPhee*

Year of Study: *2003-2004*

Scheme: *Information Assurance and Computer Crime*

University of Glamorgan

Prifysgol Morgannwg

School of Computing

STATEMENT OF ORIGINALITY

This is to certify that, except where specific reference is made, the work described in this project is the result of the investigation carried out by the student, and that neither this project nor any part of it has been presented, or is currently being submitted in candidature for any award other than in part for the M.Sc. award, School of Computing from the University of Glamorgan.

Signed...........……………………………………………………...

Contents

Abstract

As digital media becomes cheaper, and the capacities greater, the task of the digital forensic investigator will become more difficult. Huge disk drives mean that investigators may have to sieve through hundreds of gigabytes of data in the pursuit of evidence. There are only three ways to solve this problem. More processing power, clever filtering and distributed analysis. This project sets out the building blocks for a multi-user forensic system using a client server paradigm.

The project consists of 12 prototype programs intended as proof of concept that evidence can be held on a central store, that it can be controlled, manipulated on the server and finally accessed remotely.

We find that the building blocks of an open, distributed, system are available and if used, could help greatly increase performance in the next generation of digital forensic programs.

Acknowledgements

I would like to take this opportunity to thank Dr Iain Sutherland for his assistance while working on this project. In fact, the use of XML was his idea in the first place.

I’d also like to thank my wife Val for bearing with me when I started to panic!

Figures

Chapter 1, Introduction

1.0 Context

The development of PCs in the last 25 years has been remarkable. If we look more deeply, it seems that some aspects of their development are more successful. When we attempt to compare computers it is usually in terms of their hardware. Speeds are higher. Capacities are greater. Software is rarely mentioned. When we look at hardware in more detail it is clear that not all components have developed at the same rate. Processor speeds have improved a lot. The original PC was 8 bit 1.6MHz today we work on 3GHz 32 bit processors. The original PC had a 160k floppy disk drive; today we have 250GBytes hard disks. Each of these is an improvement of the magnitude of millions of times. Disk data transfer rate have not improved as well. The original IBM PC had a data transfer rate of 1MB/Sec, IDE Disks have a transfer rate of 3.3Mbits/Sec, and even the very latest Serial ATA III interface transfers data at only 600Mbits/Sec (SearchStorage.com, 2004). This is a fractional improvement in the range of hundreds not millions when compared to disk capacity. This then is the problem for digital forensics in the next few years. The evidence on huge disks is locked in by the bottleneck of data transfer speeds. One solution to this problem is to move away from PCs when analysing data onto larger mainframes with greater abilities to cache data in huge RAM memory. To do this forensic software would have to adopt a client/server configuration where data is stored, and for the most part analysed, on the mainframe and then distributed to investigators utilising the very successful GUI interface of Windows.

I have been writing software on PCs for 25 years. Since 1980, I have created more than 50 systems primarily for small businesses. They range from invoicing and stock control through communications and often include importing and exporting data between programs. Early programming languages on PCs required the programmer to create and manage their own random access or sequential data files but when Dbase II was introduced in about 1982 the situation changed. From this point, data was more often stored in formal databases. This opened up greater power to build applications and share data without the need for conversion. Before 1985, most systems were written for single user access but as networks became more common, inevitably multi-user systems were developed. This transition was made easier by the use of databases. More recently, the Internet has had a huge effect on the use of computer. Business users have come to expect their multi-user systems to work across the Internet. This is currently the major development area in PC software with VPNs becoming as common as LANs in the small business sector. The .NET technology is clear evidence of Microsoft’s interest in this area but there are many other technical solutions available.

In search of business solutions, the author has explored the use of database servers available on the Internet and providing a program interface using a web browser, typically Internet Explorer. Specifically using Linux, MySQL, PHP and Internet Explorer some effective user interfaces have been created. It is the author’s opinion that lessons from this development can be applied to the development of the software used for forensic analysis.

1.1 Scope of the Dissertation

The ultimate objective of this project is to develop a small forensic application that demonstrates the key requirement of forensic software in the near future. It must be made clear that this project is not intended to produce a full competitive product. More realistically, this dissertation should be view as an exploration of the use of Internet technology as a base of a client/server forensic tool. We hope that most problems are within the capabilities of the author but perhaps not. These may be left for future for others to develop if they think it worth the time.

1.3 Aims

The aim of this project is

- Explore and evaluate the use of web database technology in the creation of a multi-user, client/server forensic tool, to be used over a wide-area network that will, as a whole, enable future forensic software to deal with large media.
- To consider the merits of a centralise system where highly skilled professionals control the evidence and lesser skilled operatives have remote assess to the evidence.

1.4 Objectives

The objective of this project is

- To create a series of prototype programs to establish whether the data available in a 'dd' disk image can be delivered to remote users in a form that allows forensic analysis of the data.

1.5 Usefulness

This project may be of use to anyone developing forensic software in the near future. It attempts to lay the basis of a client server type structure to distributing forensic data across a wide area network.

1.6 Structure of the dissertation

1.6.1 Chapter 1, Introduction

Chapter 1 introduces the subject area of the project. It places the project in the context of the authors commercial experience over the last 25 years and sets aims and objectives for the project.

1.6.2 Chapter 2, The future of the digital forensics

In this chapter, we highlight the rapid development of computers and suggest that, with the tools currently available, the forensic investigator will be loosing the battle to keep up with the demands made upon them to investigate digital media. We find there is almost no research published on this matter. It may be because many in digital forensic simply don’t foresee this problem, or at least not a solution.

1.6.3 Chapter 3, A digital forensic methodology

Digital forensics lives in the shadow of a body of general forensics that has had many centuries to evolve a formal, accepted methodology. Since the idea of using science to solve crime developed in the late 1800s, there has been ample time to make mistakes and refine thinking within the subject. Digital Forensics is scrambling to catch up. Individual tasks are well documented; we will look at the works of Warren Kruse and Eoghan Casey to see the stages of investigation. Perhaps in this way, we can develop a design brief for a proposed system. This chapter considers the tasks within a forensic investigation to build up a design brief for the prototypes.

1.6.4 Chapter 4, Wide Area Network Technology

When designing software we must take different approaches depending on the nature of the connection between processor and data. When data and processing are homed on the same host and the links between the processor, RAM memory and disk memory are very fast and reliable, stateful connections can be made between processes and data. When the data source is remote, the link may not be so reliable, as when a wide area network uses slower public communications lines, and so it is better to use non-stateful protocols. Ultimately these non-stateful protocols provide a more robust link. In this chapter, we consider wide area network links and choose an appropriate data transfer protocol for the design.

1.6.5 Chapter 5, System Design

In this chapter, we choose a design strategy and set the design aims and objectives in more detail. Resources and constraints are listed and the fundamental structure of the system is devised. We choose overall system structure and data structure. Finally, we briefly discuss access control.

1.6.6 Chapter 6, Prototype Designs

This is the heart of the project. Programs 1 to 4 build the basic blocks of selecting media, mounting it and assembling a MySQL database of evidence.

MD5 digests are used to reduce noise and highlight suspicious files. Program 5 to 7 develop the idea of delivering evidential data by using XML. Programs 8 to 10 prove that binary data can be extracted from, at least, a FAT12 image. Programs 11 and 12 gather these ideas together and demonstrate a client program written in Visual Basic using XML as a data source.

1.6.7 Chapter 7, Testing

Because the results of a forensic investigation could eventually deprive an individual of their freedom, and in some countries their life, the standards for testing forensic software need to be higher then the test standards of the original application or operating system software than was used to create the evidence. In this chapter, we consider how others test software and devise a group of tests for the prototype software developed so far. This leads on the idea of a self-assuring approach to the development of the software where error checking is built into the final design.

1.6.8 Chapter 8, Evaluation

In this chapter, we evaluate the successes and failures of the project to meet the design aims and objectives set in chapter 1 and chapter 5.

1.6.9 Chapter 9, Possibilities for Future Development

As this project was always about basic technology developments and formulating a fundamental structure for future programs, it is not surprising that there are many possibilities for future development. In this chapter we consider just some that have been brushed against in the course of this project and could themselves be interesting future development work for this writer or others.

1.6.10 Chapter 10, Conclusions

We finally draw this project to a close by reviewing all that has been said over the past 9 chapters.

Chapter 2, The future of the digital forensics

2.1 Introduction

In this chapter, we highlight the rapid development of computers and suggest that, with the tools currently available, the forensic investigator will soon be loosing the battle to keep up with the demands made upon them to investigate digital media. As we will see, there is shockingly little research in this area.

2.2 Lack of Research

When this project started in July 2004, despite extensive research, no material was found directly related to solving the problem of forensic software performance or wide area distribution.

Many commentators pointed out the problem. Lisa Bratby in her article in Police Review, 30th April 2004 “Hi-tech moves” quotes Detective Sergeant Geoff Fellows who heads up the computer crime unit in Northampton says that his department is ‘immensely busy’. He sees workload increasing because of the expansion in the domestic use of computers, with many homes having more than one computer, and the size of hard drives and storage media such as CDs and DVDs continually growing.

In ‘Forensic Computing’, Tony Sammes considers the issue of Bigger and Bigger Disks in Chapter 8, Looking Ahead (Just a Little Bit). Writing in 2000 he quotes a Time computer with a 27GB drive representing problems for the forensic investigator. Now, in 2004, it is not possible to buy a hard disk smaller than 40GB. But all he says is “Unfortunately, we have no easy answers.” In the same chapter he addresses the issue of legal privilege with regards the protecting of data contained on a hard disk but not relevant to the specifics of the case. He speculates about solutions but offers no answers except to suggest that granting custody to a central authority that issues evidence on a need to know arrangement.

Perhaps the lack of research is because many commentators are from a Law enforcement background, are aware of the problem area, but lack the specific computer knowledge to provide technical solutions.

As a last attempt to find new research, an additional search was attempted in September of 2004 just 2 weeks before the completion of the project. This did yield just one item. A reference on the Digital Forensic Research Workshop website led to a speaker at a conference held in Linthieum, Maryland in the United States on the 11th, 12th and 13th August 2004. In the paper “Breaking the Performance Wall: The Case for Distributed Digital Forensics”, Vassil Roussev and Dr Golden C. Richard III propose the use of a 64 node Beowolf Cluster to tackle the need for power (see chapter 3). Dr Richard’s email is in Appendix A and confirms that

“We too see a huge void in this [research] area--I don't know of any other work (aside from yours, now). “

2.3 Rapid growth of the digital world

There can’t be many things in human history that have developed as rapidly as computing, especially in the last 25 years. It seems hard to believe that it was in September 1981 (www.oldcomputers.net, 2004) that IBM released it IBM 5150 – the personal computer. At the time, it had 16k of memory but could be expanded to a massive 64k; 16k of memory was about £92. The original PC had a 160k floppy disk; the 10MB hard disk would not be released until March 1983 in the PC-XT (www.oldcomputers.net, 2004. The 10MB hard disk was just under £1000.

At the start of this project in July 2004, about 21 years after the PC-XT was released, the public could buy a 250,000MB (250GB) hard disk from PC World for about £230. By the end of the project 3 months later, the same disk was £180. New Dell PCs are now fitted with 80GB drives as standard (Dell, 2004).

In 1977, Ken Olson, Founder of DEC said “There is no reason anyone would want a computer in their home” (Quotes for All, 2004). By 2003, almost 50% of households in the UK have *at least one* PC, 50% are able to connect to the Internet and 15% connect via broadband (National Statistics, 2004). Domestic computer users are commonplace; we now have Wizz Kids and Silver Surfers. Almost every office worker any developed country has a PC on their desk.

2.4 The rise of computer crime

Computer crime grew along with the general growth of computer knowledge and usage in society. It was the likes of Kevin Mitnik who broke into Tsutomu Shimomura’s computers on Christmas Day, 1994 (Gulker ,2004) that announced the birth of computer crime in a significant way. Crime had certainly existed before but this was the first *digital bank raid*. These systems needed expert knowledge to crack and equally expert knowledge to investigate. Although the value of this single incident was set at $10 million the total value of computer crime during the 1990s was probably not that great. At this time, incidents could be viewed as individual events. In the last 10 years, computer crime has become a reality of life. The establishment of the National Hightech Crime Unit in 2001 was a waymark in acknowledging the growth of the cyber-crime in the UK. Predicting the future is always difficult, as Ken Olson can prove, but the simple observation on the potential ownership of computers and the knowledge that so many people are using e-commerce and online banking means that computer crime is here and will grow, probably faster than we can imagine.

2.5 Crime without Frontiers

The ever-expanding use of the Internet has created an environment that has a great appeal to criminal activity. Because the cyber world observes no geographic boundaries crimes can be initiated in one country with the victim in another. This presents investigators with a far greater need to co-operate with external agencies often outside of the jurisdiction of the primary investigation. Conversely, evidence from other jurisdictions may be of use in a local investigation. Criminals currently exploit the current situation; they hide behind the apparent anonymity provided by the Internet. It can be a safe crime with little or no contact with their victim.

2.6 Forensic tools

For the most part, computer forensic tools have developed from the debugging tools used by programmers. There are many similarities in the debugging task and the forensic task. Both the investigator and programmer is search for clues to solve a puzzle “What happened?”. For many years, Norton’s Disk Editor was a good example a debugging tool used by investigators. Originally intended for programmers it was excellent for viewing the contents of slack space, bad clusters and unallocated space. On a floppy disk, it was ideal. On a small hard disk, at the time perhaps 20MB, it became a little more time consuming. Today on a 200GB drive, unless the investigator knows exactly which cluster to look at, it is pointless.

In fact, most tools become tediously slow when confronted with huge media. Even formatting a 128GB, drive can take 24 hours. Imaging a 128GB drive can take the same amount of time, and more. This is primarily a function of data transfer speed. Disk capacities have increased from 160k in 1981 when the IBM-XT was released to 250GB available on the high street, in PC World, in the summer of 2004. That’s an increase of 1.5 million times. In the same period, data transfer rates have only changed from 3Mbits/sec to 16Mbits/Sec.

2.7 The key problems in the near future

Quantity of Data:	It takes hours, if not days, to image and reproduce images for investigation. Multiple hard disks have to be forensically cleaned and images written to them.
Processing time:	Interactively searching through 128GB of data for text strings is very time consuming. Even a simple text search can take hours with regular

	expressions it can take days. Viewing hundreds or even thousands of images can take just as long.
Inextricably linked data:	There is currently no effective way of masking sensitive data while maintaining the forensic integrity of the remaining data.
Distribution of data:	Multiple copies of sets of DVDs have to be created. Even with compression, that technically is altering the data (infringing ACPO principle 1), it may take several DVDs to store the image of a large hard disk. As this is controlled data, distribution must be controlled to the highest standards. This is a major administrative task in itself.

2.8 Is forensic software up to tomorrow's demands?

No other class of user would need to search the entire disk surface for specific data in the way a forensic investigator does (see chapter 3). It is clear that the current batch of forensic tools is loosing its effectiveness. The tools still perform the task they were designed and written to perform but with increasing size of media, they take an increasingly unacceptable time to complete certain tasks. The shear scale of the task will require collaboration over wide area networks to reduce the overall time to complete an investigative task. Computer Forensic Investigation is likely to become far more of a group task. Most tools currently available are single user. They were never designed to be used by groups of investigators, so most have no provision for sharing notes with co-investigators.

2.9 Can analogues with other software help us see the development of forensic software?

It may be of benefit to see how other software applications have developed on PCs over the last twenty years.

In the early 1980s, the first accounting packages on PCs were single user. These were text-based programs running on DOS. By today's standards, these were very crude programs. They offered little in sophistications being at the time a simple recording system for financial transactions. If any post accounting calculations were needed the data would have to be re-entered into a spreadsheet. Within a few years 3rd party programs appeared, able to read directly from the data files of the original accounts package. Soon, presumably motivated by an extended market, the original accounts program manufacturers started to incorporate standard types of functions in their programs like report generators and sales forecasting functions. When Microsoft introduced Windows in the early 1980s many PC accounting programs were rewritten for Windows. Windows provided two main features; it unified the end user interface, i.e. made most programs look very similar, so that most users could use spreadsheets, word-processing and accounts without too much trouble, and it allowed data to be interchanged by the operating system. Information could now be swapped between accounts program and spreadsheet with ease by simply *cutting and pasting*.

When computers had limited storage accounts administrators were forced to *close down the accounts* at the end of each month. In this procedure the accounts program would calculate the balance to carry forward and then delete previous transactions leaving greatly reduced datasets. As disk space increased, so administrators wanted to retain their transaction history in full detail. To provide for this system, developers employed the client/server paradigm used by mainframe databases like Oracle. In this configuration, the database is held on one host (or more recently in data warehousing, more than one) whose sole task was to deal with requests for data. The data server would be optimised for sorting and selecting datasets while the client would be responsible solely for the display of the data and user interaction. IBM's SQL has become the defacto standard for this type of communication.

More recently accounts programs like Sage have extended their systems by opening up their data files via application program interfaces (APIs) to allow 3rd party developers access to their data structures. Sage have chosen Microsoft Access as their database server.

2.10 Collaborative Systems

This problem skirts near the subject of Groupware. Groupware is based on the principle that the real workplace environment can be translated into a virtual space environment with all its inherent advantages of storage, retrieval and lack of geographic limits. Collectively this software is referred to as Computer Supported Cooperative Work (CSCW). Groupware/CSCW is not new; back in the late 60s, Doug Engelbart created “oNLine” NLS/Augment Beaudouin-Lafon, 1999). This system featured most of the functions that appear in more recent systems, including hypertext linking and email. As a by-product, Doug Engelbart also invented the mouse as a HCI device to drive his software. This work was mostly seen as academic as the infrastructure for this communication did not exist. There was little point in groupware in a local office environment with high-speed network connections. In this situation, the physical environment can provide the group environment. It is only now by utilising the Internet that high-speed wide area connections are affordable by local offices that groupware can flourish. It could be that Forensic investigation software would benefit from design analysis from the CSCW point of view.

2.11 Conclusions

Because of the uneven development of disk storage and disk transfer speeds, a bottleneck has occurred that will cause the forensic investigator great problems. There may be a solution by moving away from the single user on a single PC to a group task on a distributed computer system. Some may call this groupware, a collaborative system or CSCW.

There is almost no research in this subject area of forensic tools.

This project explores the possibility of a forensic program in the client server idiom. A full system is well beyond the scope of this project but we hope to establish some practical demonstrations and from there suggest further research and development.

Chapter 3, A digital forensic methodology

3.1 Introduction

Digital forensics lives in the shadow of a large body of general forensics that has had several centuries to evolve a formal accepted methodology. Since the idea of using science to solve crime was developed by French medical jurist Antoine Louis in the late 1600s (Samples, 2004), there has been ample time to make mistakes, correct them and refine thinking within the subject. Digital Forensics is scrambling to catch up. Individual tasks are well documented; we will look at the works of Eoghan Casey and Warren Kruse to see the stages of investigation. Perhaps in this way, we can develop a design brief for a proposed system.

3.2 Case Management

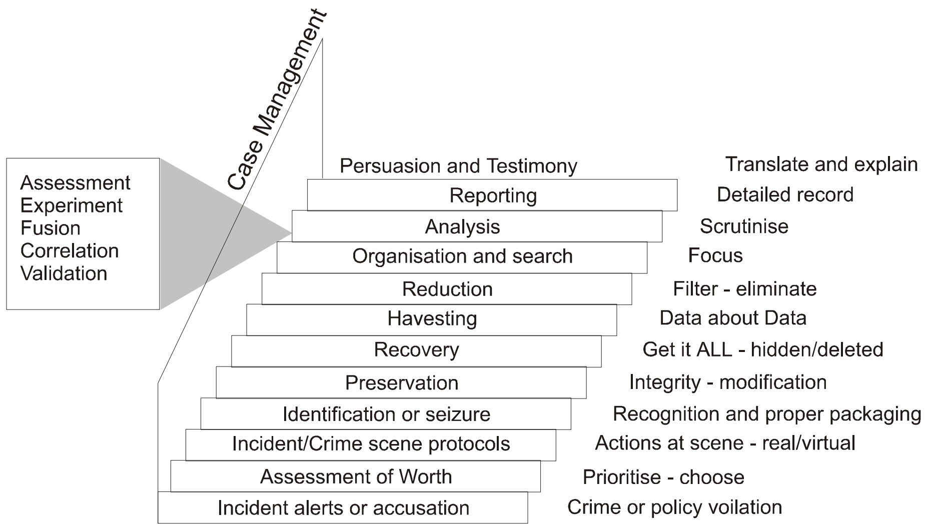

Figure 1, Case Management

This diagram, taken from Eoghan Casey and Gary Palmer, (Casey, 2004) attempts to convey the whole investigative process from incident alert through to Persuasion and Testimony. This project is only concerned with the events between Preservation and Reporting. Casey and Palmer acknowledge within their diagram that these stages can be made up of further sub-stage. They use Assessment, Experiment, Fusion, Correlation and Validation. Perhaps a more detailed list compiled from several chapters of Computer Forensics by Warren Kruse will create a more details specification.

> "Computer forensics involves the preservation, identification, extraction, and interpretation of computer media for evidentiary and/or root cause analysis."

Tasks in digital forensic analysis

- Acquisition
- Chain of Custody, Preservation and validation (MD5)
- Reduction of background noise
- Searches for
  - Known illegal evidence
  - Possible evidence
    - Encrypted data
    - Stenographed Data
    - Obfuscated data
    - Misnamed files (wrong extension)
    - Text Searches
    - Slack, Unused data areas, Swap Disk Space analysis
    - Data requiring specific software eg accounts, specialist packages
    - Tracking email
    - Reading, viewing Documents (Word, logs, pictures, email etc)
- Organisation of data
- Generating Timelines
- Reporting

Analysis and interpretation of data

Warren G Kruse II, Computer Forensics – Incident response Essentials

Figure 2, Tasks in Digital Forensic Analysis

3.3 Typical tasks in forensic investigation

3.3.1 Data Acquisition and Imaging

After the physical recovery and documentation of evidence, the first stage of the forensic task is to take copies of the original evidence. This is important, as it is a primary forensic principle that the original of any evidence should be handled as little as possible to avoid the risk of contamination during the forensic process (ACPO, 2004).

3.3.2 Chain of Custody, Preservation and Validation

All movements and handling of evidence must be recorded for future reference. This stage will include MD5 hashing of data. MD5 hashing involves processing every bit in every byte in a file or image to calculate a 32 byte (approximately 10^{12}) number that is unique to that byte pattern.

Why bother to calculate MD5 hashes?

MD5 hashing serves two purposes in forensic investigations.

1 They ‘snap shot’ the entire contents of a series of characters, in this case a disk or file. If any modification is made to the file, in any single bit, either intentionally or by accident then the result of a new MD5 calculation will yield a different number.

2 They provide a short unique identifier for the file. Rather than having to compare the entire file against a base, the MD5 can be compared; a mere 32 characters can be used as a comparison.

In this way, the forensic investigator can be assured that the evidence remains untainted during an investigation, or if it is changed, the investigator can tell (ACPO, 2004).

Limitations of MD5 fingerprinting

On the 18th 2004 the foundations of MD5 fingerprinting was thrown into doubt when a paper was presented at Crypto 2004. Four Chinese authors (Wang, Feng, Lai and Yu) have published a report of a family of collisions in MD5 (different files that could yield the same MD5 number). They report that their method can ultimately find collisions in SHA-0. Eli Biham reports that his analysis of SHA-1 is near detecting a collision of SHA-1 finger printing (http://freedom-to-tinker.com/archives/000664.html). Even with this, MD5 and SHA-1 fingerprinting still have an application in the real world. Early DNA finger printing was an improvement over the traditional fingerprinting even though there was a possibility that there was a chance of mistakes. The chance of mistakes in DNA fingerprinting were less than that of conventional fingerprinting. Recent developments in DNA finger printing have reduced the sample size needed and the error rate reduced.

MD5 digests can also be used to identify known files is that they can then be excluded from searches and so greatly reduce search times.

3.3.3 Noise Reduction.

To search or view the entire contents of a typical disk found on a computer is a time consuming business. To reduce this time we can attempt to reduce the quantity of data to search. There are several methods of reducing noise but identifying files that are known to be part of the operating system or applications and them eliminating them from any further searches or selections is by far the most effective. We can use the MD5 hash of each file for this purpose. An extensive set of MD5 hashes can be imported from external sources such as NIST or the National Drug Intelligence Center (NDIC) Hashkeeper program datasets or the investigator can develop their own. When a match is made, a filter can be set to suppress of highlight the files as appropriate. NIST report that 60-75% of operating system files and 80-95% of application files can be eliminated this way (White, 2004).

3.3.4 Organisation and Search.

This phase is intended to focus the investigation to specific items. It seems sensible to first start looking for obvious illegal items. The MD5 search approach can be extended to highlight not only items that are known to be 'good' but also those that are known to be suspect. Potentially suspicious items can also be highlighted by checking their file type signatures.
For example

File Type	Begins with
GIF	/47 /49 /46 /38 /37 /61
TIF	/4d /4d
JPG	/ff /d8
DOC	/31 /BE /00 /00 /00 /AB

Figure 3, Examples of File Signatures

(Sammes and Jenkinson , 2002) (Oliver, 2004)

Most standard file types have unique patterns of bytes within the file that can be used to identify the structure of the contents. In and effort to obscure incriminating information the file extension can be change. For example an incriminating Word document may have its normal extension, .doc, changed to .xls. In this way, the image will not open using Excel, giving a ‘file format is not valid’ error. By changing the .doc extension to .xls the suspect may effectively exclude this file from any search that may attempt to limit it s coverage by selecting only *.doc files. This function is not included in these prototype programs.

3.3.5 Analysis. Assessment, Experiment, Correlation, Validation

This is the Scrutiny stage when notes and cross-references are made. This will involve a chain of searches and compilation tasks. Further manual filtering will occur to further reduce the noise level. At this stage, the discoveries of several investigators may need to be drawn together. This may be hampered when the investigators are not in geographical proximately to one another.

3.3.6 Generation of Time Lines

Often investigation are centred on the exact sequence and time that events occurred. Using the Modification, Access and Creation times (MAC) of files data use can be built into a timeline representing the chain of events. These could be the creation time of file in the Internet browse cache directory.

3.3.7 Creating and audit log

It is essential that as well as keeping chain of custody information a full audit log of any investigative work is kept to allow the recreation of the investigative process by other investigators either for corroboration or defence.

3.3.8 Reporting

The ultimate objective of all this work may be to present evidence before a Court. It aids clarity and encourages an air of professionalism when the reporting is in a constant format.

3.4 Key features of existing software

The idea of Forensic Computing to detect and counter crime developed in the mid 1990s. It may be fair to say that the coming of age with regards forensic tools was when Guidance Software, from Pasadena, California, release EnCase in 1998. In some respects EnCase did nothing new, what it did was integrate all the relevant tools in one package together with the ability to verify the consistency of evidence and report on findings in a consistent way that could stand-up to judicial scrutiny. As is often the case, the primary developer has the opportunity of a 'territory grab' and so EnCase has set Guidance Software as the leading Forensic Computing software provider in the world. No assessment would be complete without including EnCase and it is as good a place to start as any.

3.4.1 EnCase

There are a number of key features to the way Guidance have designed EnCase. (From John Patzakis, 2002.)

Microsoft has been astoundingly successful in establishing themselves as the number one provider of operating systems for PCs. In fact many people may never see a computer running any other operating system, or at least not realise a computer was running something else, but Windows is not the only choice. In fact, many may argue it is not the best choice for forensic work. It has a major advantage in that it is so well known. There can be no doubt that Microsoft's success with Windows is largely down to the use of the Desktop environment and the mouse. It is also fair to say that the vast majority of the computers encountered will be running some version of Windows. Perhaps Guidance chose to write EnCase on Windows for marketing reasons. When 'selling the system' the majority of the audience in 1998, as indeed today, would be Law-Enforcement officers who surely would little experience of other operating systems. Selling EnCase in Windows is pushing at a half open door. Windows has several drawbacks for the forensic investigator. All versions of Windows require that all mounted file systems are write enabled by the operating system. Windows insists on creating a 'recycled bin' on any drive it encounters. This is contrary to the principle of the forensic investigation not affecting the evidence (ACPO, 2003). Users of EnCase have to resort to extra hardware like a FastBlock to protect their original evidence (ICS, 2004). Because EnCase does not, or rather is unable to, mount the evidential file system on the operating system file system they will have had to write their own drivers for each file format. Guidance Software may argue better this than accept other peoples work but as EnCase is not Open Software the forensic community is left to trust that Guidance have done this properly. As it cannot be mounted directly this allows guidance to modify the evidence file with compression and MD5 checking within the file. Strictly this means that Guidance are departing from the ACPO principle 1 - that of non-amendment of evidence. Presumably, they fall back on principle 2 - that any amendment shall be made by a qualified individual who can account for their actions. Guidance publish their file structure and would maintain that their software is ‘a qualified individual’. Guidance have introduced a CRC checksum every 32kb within the evidence file saying that it secures the evidence in a more granular way. It may be that Guidance has chosen to verify each 32kb section because they do actually write to the evidence file, but only in the section allocated for case information at the start of the file.

In terms of features, EnCase is very effective in providing a suit of Investigatory tools. The Investigator can recover deleted files and folders, perform file signature analysis as well as MD5 hash calculations. EnCase can then apply filters to reduce noise and highlight evidence placing these into record sets for further investigation. Evidence can be viewed as timelines and in gallery format to aid rapid viewing of images. Files, within filters, can be searched for text in a variety of formats and Unicode extended text formats are supported for non-Cyrillic languages and alphabets. The searches also support Unix regular expressions to enable pattern matching to locate, for example credit card number patterns. Finally, EnCase has a scripting language to allow users with less experience to perform complex tasks written by investigators that are more experienced or even professional programmers. For the investigation of Windows evidence, which is currently the bulk of evidence now, there is a

Windows Registry viewer. All investigator activities are logged in the audit log and reporting is semi automatic across all functions.
As a system Encase takes the ‘fast start-up – slow search’ approach. Rather than spending, perhaps, hours indexing and sorting information before the evidence is investigated Guidance have chosen the launch the program as fast as possible at the expense of slower search times in operation. In addition, EnCase is actually single user. Any notes made by an investigator could overwrite notes made by another investigator working on the same image file across a network. EnCase does run on a workstation with the evidence available on a shared drive on the network server but as this means that the evidential data has to be called to the workstation for the program to perform the search criteria test. As this could mean the entire evidential image has to be called to the workstation, it has a huge impact on the search speed. EnCase performs searches in the background but this processor time is stolen from other tasks.

3.5 EnCase Enterprise

EnCase Enterprise represented a significant increase in Investigatory power by extending the abilities of EnCase across a LAN or a WAN. Based on a secure server (SAFE - Secure Authentication For EnCase), the investigator uses a version of EnCase called the Examiner to examine the contents of any computer, authenticated to the SAFE, running the EnCase Servlet. In this version, the Examiner can interrogate the evidential computer in question without interrupting the operation or use of the computer. This does raise serious issues of privacy but these are not relevant within the boundary of this project.

Files can be preview and ultimately downloaded to the SAFE for further investigation and retention for evidential purposes. Ultimately the entire disk could be downloaded although the practicality of this is limited by the bandwidth available but a moderately large disk could be imaged overnight or at the weekend over a fast network. File encryption is an increasing problem for the investigator, EnCase Enterprise allows the investigator to view the files after the suspect has entered or activated a decryption key, as they are using them. It also allows the investigator to 'snap shot' volatile information in RAM, port information and live registry analysis.

3.6 Brian Carrier's Autopsy

3.6.1 Introduction

Autopsy and the Sleuth Kit have been developed as Open Source experiments. As a result, it is possible to analyse its workings in much more detail. It is very important to make accommodation when making comparisons with EnCase. EnCase is the product of a large company, Autopsy by a handful of educated enthusiasts in the tradition of Open Source Software. However, lessons can be learned from both.

The Sleuth kit was developed over a number of years by Dan Farmer & Wietse Venema as a set of forensic utilities with assistance from @stake and was originally called The @stake Sleuth Kit (TASK).

From the start The Sleuth kit requires a high standard of computer knowledge to install. The utilities are written in C, Autopsy is written in perl. To use the Sleuth Kit, the user needed access to, and the knowledge to control, a Linux shell prompt. In its current incarnation, the Sleuth Kit is obtained by downloading from Brian Carrier's web site, www.sleuth.org. It then needs to be gunzipped, extracted from a tarball and then compiled and installed. They would then download and install autopsy. The Sleuth Kit and Autopsy normally run on Linux systems but can run under Windows with Cygwin . Cygwin is a Linux 'emulator' for Windows. This adds a further level of complexity and so it is better to work on Linux if possible. Autopsy is dependant on a lot of its functionality from The Sleuth Kit.

Brian Carrier developed Autopsy as a graphical interface for The Sleuth Kit making it available through the more accessible interface of a Web Browser. Some may argue that the digital investigator needs this higher knowledge but if the vast increase in digital evidence does happen the investigative forces may not have the luxury of being this fussy and have to utilise lesser-qualified staff. If Microsoft has done nothing else, they have made software installation easier. Brian Carrier has decided not to use existing web servers like Apache but instead he has written his own server function and run it from port 9999, not 80 as would be normal for web data. He has done this of ease of installation not security reasons (see Appendix A). This enables Autopsy to be run on the same host as a web server without inference and avoids the configuration of an Apache server. The software can handle multiple users but this requires that additional copies of autopsy are run and that they use separate ports i.e. 9998, 9997 etc. The standard software has been written to work on the ‘localhost’. This is presumably security measure to control ‘outside access’ but there is a command line parameter available when invoking Autopsy to allow access from a specific IP numbered host. This invocation requires a variable and long URL to be entered on the host. For example:

http://serverhostname:9999/289465758406108856/autopsy

each time the server is started it generates a new random number as a sort of cookie. This ensures security. Again, this can be overridden by the command line invocation.

Unfortunately, because Brian has chosen to write his own http server function there is no SSL transmission encryption available but this would not be needed in a closed environment.

The program starts by presenting the investigator with an opportunity to create a ‘Case File’ or to open a new Case. Once open the investigator can open Disk image files of various formats and by doing so associate them with the particular case.

It is not reasonable to expect Autopsy to match EnCase command for command but most of the basic functions found in EnCase are present.

- The ability to sort, filter and then view files in an appropriate format i.e. graphic, Hex or Ascii. Data can be followed by logical file structure or by disk cluster.

- The ability to create MD5 hashes and then compare them with an authoritative database, e.g. NIST.

- The ability to view a timeline of file creation and modification dates.

- Extensive logging of investigator activity, in plain text files as is the tradition in the Unix environment.

- Text searching

- A lot of data generated like the number of files found on the same day during the timeline analysis is saved as text files within the directory structure of a case.

The Sleuth kit lacks

- A Windows registry viewer

- A scripting language

One important feature already mentioned is that Autopsy is a ‘front end’ for Weitse Venema’s Sleuth Toolkit. There is no reason then that Autopsy could not be extended as a front end for many other utilities created by other software writers.

Autopsy does represent an attempt at a wide area network forensic tool. It is limited by the use of plain HTML available in a browser. As Brian Carrier says, he is not a “GUI man” and the project has “gotten much bigger then it was originally intended to.”

3.7 Forensic Tools with XML

By Yiannis Koukouras and Emmanouil Vlastos of the University of Glamorgan

Published in October 2003 this is worth a short note. Yiannis and Emmanouil have addressed the issue of defining a forensic source in XML format. In addition, they have written an Analysis Viewer in Java.

The software can be downloaded from sourceforge.net and saved in a folder of the users choosing. The forensic data is held in a text file on the local host. The viewer opens the XML file much as Microsoft Word opens a document. The viewer displays the contents of the XML file in a manner more familiar to users of a GUI environment like Microsoft Windows, Gnome or KDE. This program does not processing other than to parse the XML and display it but it does demonstrate the use of a non-platform specific client program written in Java (see Chapter 4).

3.8 Power computing applied to digital forensics

This is an opportunity to draw in the work currently being researched by Vassil Rouissev and Dr Golden Richard of the University of New Orleans (Rouissev & Richard, 2004). Dr Richard's background is in supercomputing and so they have approached the problem rather differently. They suggest that any approach that intends to solve the problem by processing disk-based images will have to pay the price at some stage. They observe two fundamental approaches. Encase does relatively little pre-processing so start up is fast but subsequent accesses and searches are slow. Their other favourite package, the Forensic Toolkit (FTK, 2004), takes much longer to start with initial processing building an index file of links but subsequent searches are faster. Their contention is that the bulk of the problem will not be solved by clever programming and that the best solution is distributed processing. Subsequently they have used 8 of the 72 nodes in the Gumbo-72 Beowolf cluster at the University of New Orleans. Their results are startling. They spilt a 6GB images across 8 machines. The images are held in cache memory and searches are concurrent. Searches that took 8½ minutes on a 3GHz PC take 27 seconds on the Beowolf cluster. As they point out this approach is not too expensive either. The 8 nodes cost little more than the cost of 8 workstations on a network. In their Discussion section they suggests that a parallel processing strategy to build indexes and other thumbnail type data in the spare time after the evidence image is loaded but allowing the user to interact while this processes is on going.

The team formulates a short list of requirements for a high power forensic system.

- Scalability
- Platform-Independence
- Lightweight – Efficient distribution of data
- Lightweight - Easy Administration
- Interactive
- Extensibility
- Robustness

It must be left to the imagination as to the results they might get on something like Virginia Tech's Terascale System X (Virgina, 2003).

3.9 CTOSE XML format to package an item of evidence

Although we do not a have access to the structures, the European funded group The Cyber Tools On-Line Search for Evidence project - CTOSE (CTOSE, 2004) have developed some XML DTDs for transmitting parcels of evidence. If this project developed into a larger scale project, these definitions would probably be the basis of an XML transmission (see chapter 4).

3.10 A Critical Appraisal of these systems

EnCase have a huge market force behind them and no doubt have significant backing from the Military, Government and Law Enforcement Agencies in the United States. It is not surprising then that EnCase is a closed system as it is a proprietary commercial product. There are facilities to import and export data but these are limited. Guidance Software's decision to use a proprietary file format may be highly commercially motivated. It is possible that use of the format by a third party will be countered with legal action to defend its intellectual copy write over the format or action under the Digital Millennium Copyright Act (DMCA, 2000). EnCase does not appear to have the ability to share its data or to link to external data-sources other than EnCase Servlets in the Enterprise version. Guidance Software has provided many facilities within EnCase. The Windows GUI is a suitable environment for this but in an effort to deliver a multi-function tool in one package the user interface has resulting so many tabs and clicks and highlighting icons that the investigator can become confused when using the package. Brian Carrier’s Autopsy is restricted by the use of simple HTML, Koukouras and Vlastos’ Java forensic tool is an excellent example of client programming but does not actually link to a data source. None of these programs addresses the issues of privacy in the evidential data. When the first seizure order were given to seize computer data it became clear that if the seized computer hardware contained data outside the bounds of the seizure warrant then the seizure could be argued as being illegal. The Criminal Justice and Police Act 2001 s50 (3), *Additional Powers of seizure from premises* extends the seizure powers to inextricably linked data and equipment. Under this section the authorities can seize the entire contents of a hard disk, for example, but they must not look at data that does not directly concern the investigation. This is a work around because the current technology has no way of separating the data without altering it. It would be preferable if future technology could mask out data outside the bounds of a seizure order.

3.11 Conclusions

In this chapter, we have reviewed the main features of EnCase, Autopsy, XML Forensic Tool and a Distributed Forensic tools. The features highlighted in this chapter are key features that should be considered when designing a forensic tool in the future.

Chapter 4, Wide Area Network Technology

4.1 Introduction

When designing software we must take different approaches depending on the nature of the connection between processor and data. When data and processing are homed on the same host and the links between the processor, RAM memory and disk memory are relatively fast and reliable, stateful connections can be made between processes and data releying on constant connections. When the data source is remote, the link may not be so reliable, as when a wide area network uses slower public communications lines, and so it is better to use non-stateful protocols. Ultimately these non-stateful protocols provide a more robust link. In this chapter, we consider wide area network links and choose an appropriate data transfer protocol for the design.

4.2 Low cost communications

Communications used to be very expensive. This was primarily because the original approach was to connect hosts with specific hard-wired connections. Clearly, this was expensive because each new connection needed actual physical work done to make the connection and after that, it was devoted to a specific task. The number of connections increases exponentially based on the number of hosts. During the 1970s and 80s so called packet networks were developed (Hobbes, 2004) that used routers to connect hosts in a logical manner. Each host on the network would only need to be connected to a local router to join the wide area network. The routers dealt with the traffic through the network to deliver the information to the 'other end' of the communications link. As well as the hardware, network communications protocols like TCP/IP were developed. This then is the basis of the Internet. Routed networks have dramatically reduced the cost of networking. Now 50% of all households in the UK regularly connect to the Internet via lost cost phone lines and more recently broadband. Only a few years ago mainframes were connected at speeds considerably less than an average domestic user can expect today.

4.3 Link Reliability

There is one major difference between a LAN and the Internet. Across the Internet, links are not as reliable as they are on a LAN. All higher protocols built upon the TCP/IP base take into account the possibility, or likelihood, that the link will fail and reconnect occasionally.

On a LAN, a link is opened between two hosts, perhaps a workstation and a server, for an indeterminate time. It is assumed that the link will remain as long as the program and programmer requires it to be open for reading and writing. Local area networks, comprising of Ethernet network cards, Twisted pair cables, Switches and hubs are sufficiently robust that we can rely on these links to be maintained. On the Internet this is not so, although things are improving. We have no idea of the nature of the connection between any of the various links joining the hosts and routers along the line. The link may be dropped at any time. The Internet was actually designed to cater for this. A great deal of development was done under the name of ARPAnet in the 1970s (Hobbes, 2004) by the Department of Defence in the United Stated. The Department of Defence wanted a network system that would withstand the sort of failure that would be created by a large-scale military attack.

	Processor	RAM	Bus	LAN	WAN
Speed characteristics	Ultra Fast access	Very, Very Fast access	Very fast access	100MHz fast access	50MHz slower access
Link characteristics	Very high reliability	Very high reliability	High reliability	Good reliability	Lower reliability

Figure 4, Transfer Speed against reliability

4.4 A Scalable solution

This approach of catering for failure in wide area networks is very effective when scaled down to LANs and there are no great penalties in this down scaling. This provides us with one solution for all situations. It seems sensible to choose a system solution that is inherently designed for scalability across local and wide area networks.

What follows is a brief description of the key technologies available for programming in this relatively new environment.

4.5 Web Browsers and HTML

The primary human computer interface (HCI) most people experience when using the Internet is a web browser when they ‘surf the web’. Browsers use a standard method of describing the formatting of text documents by means of specific markers called tags. Typically, the tag <b> would be placed in the text to signify the start of an emboldened font and </b> to signify its end. The document would contain “The <b>cat</b>sat on the mat” and the user would see “The **cat** sat on the mat”. All of the characters used were printable (including the space character) ascii characters which were 7 bit; this was because some communications lines were only 7 bit with 1 party bit and so could not transmit 8 bit characters without extra processing. HTML 1.0 defined a useful set of about 50 tags. HTML 2, 3 and the current version, 4, have described increasingly sophisticated tags to provide web page authors with greater control over the appearance of their pages. The main advantage in using browser technology is that it usually requires no installation. In the case of Windows the browser is an installed, integral part of the operating system. If this HCI could be used it would massively simplify distribution of the user end of the system. The users access to the system through the web browser interface gives them the latest version of the software. This is the approach used by Brian Carrier in the Sleuth Kit. His software is designed to reside on the server as one copy, when it is updated, so the user has the latest version.

4.6 More dynamic web pages – scripting languages

The provision to display pictures was built in to HTML 1.0. More recently, a demand for much more feature rich multimedia has lead to creation of scripting languages to give even greater control over web page activity. Scripting languages can be run on either the web server or the client browser. Two popular scripting languages are VBScript, based on Microsoft’s Visual Basic and Java Script based on the Java language. A client side script runs on the browser and typically triggers on clients events like mouse activity. It can control many of the parameters associated with the tags in html, and so can change colours, images etc. Server side scripting languages often control web page creation depending on characteristics of client activity after submitting data to the server. For example, they may insert specific adverts depending on the 'clicks' a user has made recently. It should be noted that unfortunately many security vulnerabilities have been associated with scripting languages and a significant number of security professional disable scripting languages on their browsers. Reliance on scripting languages to provide a highly featured user interface would probably be seen as a bad design decision.

Client side scripting:	Server Side Scripting:
VBScript	VBScript
Java Script	PHP
	Cold Fusion
	JSP
	Perl

Figure 5, Client side and Server side scripting languages

4.7 Non-platform dependant software - Java

Java is the premier example of a programming language that runs on the client, can access data on a server but is not dependant on a specific client operating system. It provides the programmer with a powerful set of tools that can used to write a far richer end user interface. Koukouras and Vlastos used Java to develop their end user interface.

4.8 Platform dependant software – Visual Basic

Visual C, Borland Delphi and Visual Basic can provide the very best user interfaces but are operating system specific. It is likely that EnCase is written in Visual C or Visual C++. These languages are very platform specific and attain their high quality finish by fully integrating and exploiting the operating system i.e. Microsoft Windows.

4.9 Web Servers

Tim Berners-Lee wrote the first world wide web, http, browser in 1990 and shortly after wrote the first http server. Currently the most used web browser, by virtue that it is supplied as part of Windows, is Internet Explorer, which is now on version 6. Apache is now the he most widely used web server (Netcraft, 2004). Apache is an Open Source web server that is fully scalable with sophisticated security and load balancing features. Brian Carrier has chosen not to use an established http server like Apache but instead chooses to write his own in Autopsy. This greatly limits the scalability of his solution.

4.10 Databases

As web pages have became more dynamic more information needs be fed into them. It is possible to hold a few variations in the IF ELSE ENDIF conditional structures and in the array structures that can be found in most scripting languages but for any greater quantity some form of linked storage is required. It could be provided by a series of text files that could be accessed by #include type macro statements but again this becomes unmanageable when large amounts of data is required. Access to formal databases provide a powerful and elegant solution. Links now exist between most scripting languages and many databases.

- VBScript in Active Server Pages running on Microsoft's Internet Information Server Http server can link directly to Access and MSSQL.
- PHP can link to many databases including dBase, Berkley DBM, db++, Frontbase, FilePro, Informix, Interbase, Ingress II, MSSQL, mSQL, MySQL, Oracle 8, PostgreSQL etc.

Most of these databases are relational, in that they store structured information held as fields in a record. The records relate to each other in some defined way, perhaps product codes in a stock file. The fields within the record would all be about the same stock product.

4.11 ODBC

It is perhaps inevitable that, as with PHP, each of the connections to the different databases even within the same programming language requires different codes and protocols. In an early attempt to rationalise and simplify the mess the concept of Open Database Connectivity (ODBC) was created. In essence, database authors would create an ODBC interface program for their specific database. As this interface conformed to a set standard, the client program would only need to create a connection to an ODBC interface and may be oblivious as to the complexity of the actual native database. ODBC is successful, if a little unreliable, in local area network environments but does not translate into the wide area network because it requires a reliable link. (Eagletone and Ridley, 2001)

4.12 SGML and XML

From http://xml.coverpages.org//sgml.html

Tim Berners-Lee was heavily influenced by a development in the 1960s called Standard Generalised Markup Langauge (SGML, ISO 8879:1986). This is a method of defining a mark-up language like HTML. It is fair to say HTML could be a creation of SGML. SGML is very customisable, flexible and powerful and consequently very difficult. For our purposes, the much simpler subset version called Extensible Markup Language (XML) is of prime interest. Those familiar with HTML and its use of defined tags to format web pages will be familiar with XML. It allows the author to define their own tags and so can markup information in anyway they want but always in a structured repeatable, predictable way. A small example of XML will explain

```
<?xml version="1.0"?>
<data>
        <item id="1">
                <name>
                        A12345
                </name>
                <description>
                        A motor component
                </description>
                <price>
                        1.23
                </price>
        </item>
</data>
```

Figure 6, An example of XML

In this example, we can clearly see structure. Each opening tag (with the exception of the identification tag at the top) has a closing tag pair. The author chooses the tag names. The indentation is only for human readability and is not needed in XML. (Bradley, 2000).

Using XML, data is distributed in discrete packets at the request of the client. Having requested and received data the connection is dropped. If the connection fails while the transmission is running, the client would simply reinitiate a connection request.

4.12.1 Inefficiencies in XML

The XML structure could be used to store the data on the hard disk of the host but it is very inefficient. Even in the small example about the storage overhead, represented by the tags is high. Of 157 characters, only 33 are actual data, the other 124 are tags. It is more sensible to store the data in an established, proven, high performance database optimised for the task e.g. MySQL and use a programming language to dynamically create XML as and when needed.

4.12.2 XML and e-GIF

XML has been adopted as the key standard for data exchange within the government's e-Government Interoperability Framework (e-GIF) (Cabinet Office, 2004). This is an essential component of e-Government Strategy and sets out the policy and standards for interoperability of electronic data across the public sector. As a large number of the potential users of any system built upon the principles devised in this project are likely to be public sector, law enforcement or military then it is sensible to try to take a design approach that is sympathetic with the objectives and methods of e-GIF.

4.13 Conclusions

We have considered the nature of connections over wide area networks, namely the Internet, and seen that a client-server configuration is the best solution to overcome its weaknesses. In choosing a data transfer method to deliver data to the user, we have considered HTML and the web browser and acknowledged that it is too limited to provide a sophisticated user interface.

Enhancement of the HTML, browser combination with client scripting would generate security concerns within the potential user groups. Ideally Java seems to combine the HCI control we desire. Given this, XML is the current choice as an open protocol to transfer data.

Chapter 5, System Design

5.1 Introduction

In this chapter, we choose a design strategy and set the design aims and objectives in more detail. Resources and constraints are listed and the fundamental structure of the prototype system is devised.

5.2 The Design Approach

There are in broad terms perhaps three approaches to writing software (Smith, 1991).

The Analytical approach	where a study is made of the users requirements, software developed, tested and introduced to the final user.
The Engineering approach	where everything is defined to such a degree that nothing can go wrong, or at least, if it does nobody will admit it.
The Prototype approach	where the developer works far more closely with the end user. They are not afraid to ditch solutions when they are not fruitful when, along the way, solutions that are more promising present themselves. These new solutions often only reveal themselves because of the failure of previous tried attempts.

The later sounds more promising in terms of final solution but runs the risk of encountering management problems as it may seem to be rambling out of control. However the first two approaches are perhaps more risky because they both adopt more a more rigid definition of acceptable and are so inflexible to the developing needs of the user.

Prototyping especially lends itself to small projects, though Smith argues that large projects would benefit from prototyping, and is ideal for this type and stage of project. After considering the nature of this project, the time-scale and resources the prototype approach seems most promising. The fundamental issues addressed in this project are too technical to involve the end user. The next stage, of creating a real user interface, would require user input.

5.3 Aims

In the next few years, as disk media becomes larger and there is more of it to examine, we anticipate that the current set of forensic tools will start to encounter problems. There are several ways to improve performance; hardware, software and system. This project makes moves on software and system fronts. The hardware use is a standard PC (see 5.5 – Resources) but instead of running Windows with its processor expensive GUI it is running Linux. Also not all processing is completed on a single PC. Instead, we have adopted a client/server approach. This allows a Linux server to be devoted to searching, indexing etc leaving Windows clients to display the data subset. In this way, we hope to establish a new architecture for forensic programs that will enable larger media to be dealt with efficiently.

The resulting system will have a number of key features:

Most processing should take place on the server.

Building indexes, MD5 hashing and filtering should all take place on the server. The client should, for the most part, only be used to display data.

Central Control over evidence

This project asserts that it is better to have very highly skilled and trusted personnel in charge of the original evidence but these personnel are going to be limited in number. Central control of evidence will provide more secure environment for the original evidence and greater control over sensitive data.

Wide Area Access

From the central evidence store, many more people, with different interests, will need access to the evidence. This implies that there must be access control to the evidence not only at a whole image level but if possible, at a sector or cluster or file level.

Much greater processing power

Although this project uses only a standard PC running Linux it is anticipated that much more powerful hardware will power the server. Perhaps a mainframe or Beowolf cluster (see Dr Richard's work chapter 2).

5.4 Objectives

This series of prototypes is intended to prove several unique fundamental functions required to build a wide area network, client/server, Co-operative forensic tool. The prototypes specifically address several issues that would be key to this problem.

- To enable control a library of binary images of evidential disks
- To control access to the images
- To pre-process the disk files to reduce 'investigative noise'
- To analyse and make sense of a disk at a binary level and display the results.
- To convey this information to the user in the form of a simple hex viewer like Norton Disk Editor

5.5 Resources

All the development was done on what would be a small host system. The system is:

- 2GHz PC, 40GB Hard Disk, 512MB RAM, 2MB Internet Link
  This is small when we compare it with Dr Richard's Beowolf solution but it is all that is available.

- Redhat Linux 9.1
  The core Linux operating system is now distributed in many packaging arrangements; Redhat is surely the largest. Although version 9.1 is actually past the ‘end of support date’ the latest incarnation, Fedora, is promoted as bleeding edge technology rather than just cutting edge. This could be problematic and so version 9.1, despite the lack of support, is chosen for this development. Linux can run on a huge range of hardware platforms and is ideal as it is scalable from a Pentium III through to mainframes and grid computing.
- Apache 2.0
  The Apache web server is the most widely use web server used on the Internet. Currently about 68% of all web sites are run with Apache. The nearest second is Microsoft’s IIS with about 25% (Netcraft, 2004). Apache 2.0 provides sophisticated access control and load bearing configuration facilities and is scalable.
- MySQL 3.23.58
  MySQL is a very well known relational database. It is the world’s most popular Open Source database (MySQL.com 2004). It is used by NASA, HP, Yahoo and Google amongst others. It provides a stable, high speed, scalable solution.
- PHP 4.2.2
  PHP is a widely used, general purpose, open source, scripting language that is especially suited for Web Development (PHP, 2004). It has extensive capabilities to connect to various databases.

It should be noted that all the development tools used in this project are open source. In saying “The proper way to test forensic tools is by using an open method” (Carrier, Open Source Digital Forensic Tools, p4, 2003) Brian Carrier acknowledges that, from a legal point of view, is desirable for transparency in dealing with evidence and this ethos is carried through in this project.

5.6 Constraints

- **Time**
  This is a 16-week project and as such can only attempt to establish a number of principles needed to build a full program. This project will result in a simple hex cluster viewer.
- **Resources**
  A domain, www.mscproject.org.uk, has been registered and set-up for this project and is felt adequate for any development work. It is a 2.4GHz, Pentium IV with 512MB RAM, 40GB Disk.
- **Secure Transmissions over SSL and a suitable certificate**
  If forensic data is ever transmitted over a publicly accessible network it would have to be over an SSL encrypted link. Although both secure SSL and non-secure links are available at www.mscproject.org.uk all the XML server work is on the non-secure link. This is because the certificate on the SSL link is not verified by a suitable authority such as Verisign (this would cost money!). This means that most browsers will respond by popping up a warning message on first connection. This is manageable when a user first connects with a browser but cannot be effectively handled when a program accesses an XML source under program control.
- **Programming Skills**
  Although Java would probably be best as a programming language to create the client software unfortunately these programming skills are not available within the time scale of the project. Client programming work will have to be done in Visual Basic but the principles will remain true.

5.7 Network security

Network security is not in the scope of this project but it should be noted that all the prototype programs available on the project web site run on an https, SSL secure, link. The evidential images have been placed away from the directory structure accessible to the Apache web browser and so the only way to access the data is through the database and Apache server. If this system was adopted in the real world serious consideration would have to be given to securing the server and it’s data.

5.8 Preliminary system Design

This system takes a fundamentally different approach to system design than either EnCase or Sleuth. At its heart is a partnership between mounted disk images, a MySQL database, server pre-processing and remote users accessing data with XML.

In brief, a number of disk images, created by ‘dd’, are stored in a directory. From this the user will be able to:

- Access these via a web page interface.
- Mount an image on the file system.
- Import the directory structure and file into a MySQL database.
- Calculate the MD5 hashes for these files.
- Compare these hashes with a copy of the NIST database of known file MD5 hashes.
- Access files by clicking the web page
- Run a simple hex editor on a remote workstation
- All of the above would be subject to access controls defined in the Investigator identity table.

5.9 Fundamental Model

In this design, the processing model is that of a client/server where most of the processing is carried out in the centre. The user operates a terminal. As the use of the Internet is so widespread, it seems an interesting route to try to build this using web technology.

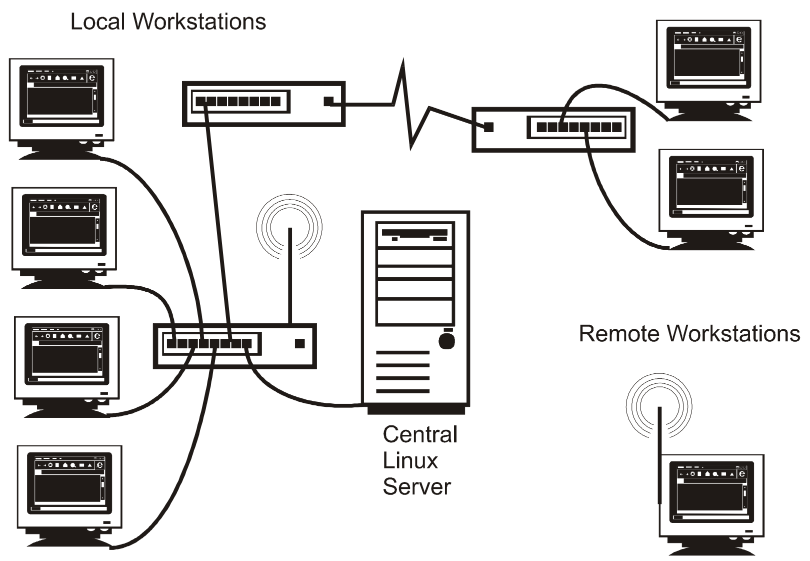

Figure 7, System Overview

5.10 General Design

The system is designed around a number of assumptions:

- That there are, and will be, relatively few very highly trained and qualified staff that can be held responsible for the safe custody of the actual evidence but many more who are capable of investigation to a lesser level.
- Labour resources are not always available in geographical convenience to the evidence.
- Evidence media will increase in size and that this will present analysts with increasing problems for storage and greater problems for transporting data in the form of DVDs etc.
- Although some evidence will require detailed expert analysis, but most would require no more than a simple but complete scan to reveal suspect material.
- That legislation like the Human Rights Act will force changes on other legislation, perhaps the Criminal Justice and Police Act s50, the Data Protection Act and the Police and Criminal Evidence Act, to restrict the access to evidential data on a need to know basis. This will impose the need for access control on an individual investigator by file and sector.
- That perhaps a solution to discrepancies in the defence and prosecution analysis of items such as viruses and Trojans may result in a single

authority, perhaps the Forensic Service or The CPS, being responsible for this analysis.

Let us imagine the scenario that all digital evidence is held in the custody of a single suitably qualified organisation, perhaps the Crown Prosecution Service or a regional investigation centre. They alone are responsible for securing and imaging the original evidence. An appropriate set of administrative programs would allow the resulting image files to be stored on a powerful central server as read only images. The system administrator would control access to this library of images via entries in a MYSQL database table.

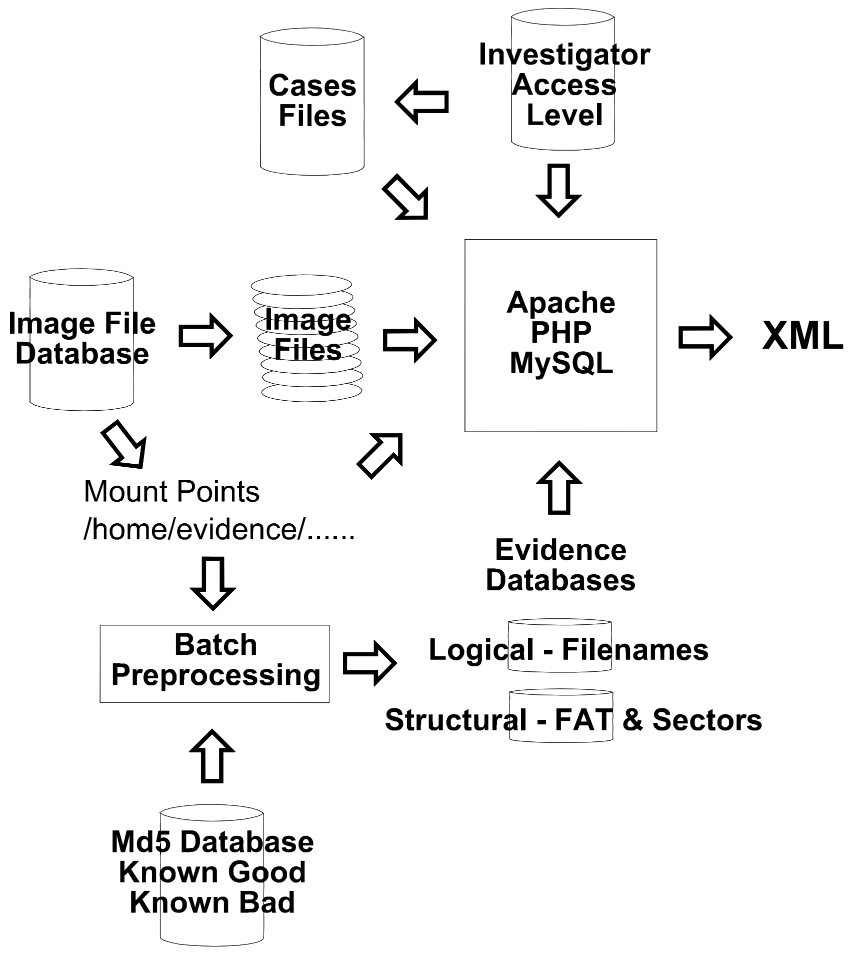
Cases Files
Investigator Access Level
Image File Database
Image Files
Apache PHP MySQL
XML
Mount Points /home/evidence/......
Evidence Databases
Batch Preprocessing
Logical - Filenames
Structural - FAT & Sectors
Md5 Database Known Good Known Bad

Figure 8, System Structure

5.11 Data Files

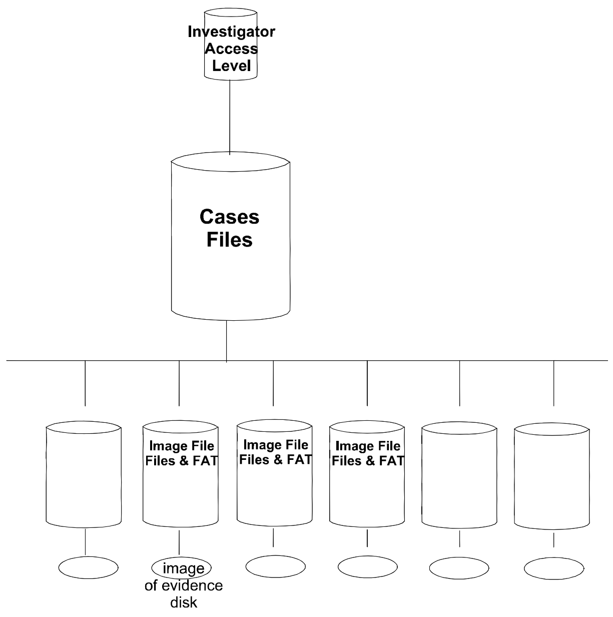

Figure 9, Case File Structure

Investigator file:	contains information about individual investigators e.g. login name, MD5 password, access rights, audit log of activities.
Case file:	contains pointers to evidence files and access level information at a whole image level. This file would contain shared note information.
Image File:	Contains several tables containing file name and sector information with access level information.

5.12 Access Control

Access to the whole system will be controlled by an *accesslevel* field held for each investigator in their specific record in The Investigator table. They would be validated against an MD5 Digest of their password and issued a time-limited cookie to maintain the session.

‘Access Level’ rights within the Investigator database will control the investigators ability to create, read, write and close/delete of cases.

Within a case file, the addition of pointers to disk image information files will be controlled by the Access Level held in the Image Information table. This contains access level information at the whole image level together with the current mount point if it is mounted.

Disk image Information files will need to contain Access Level information of each file and sector within the image.

A more sophisticated security can be employed; this should be viewed as a minimum.

5.13 Conclusions

The programs will be written as a prototype; as at this stage, we are not sure exactly how the programs will develop. The system will be built on a client server structure to remove processing from the PC to a more powerful central server. A relatively plain Pentium 2.0GHz processor running Redhat Linux 9.1 will be used as more exotic hardware is not available. However, the solution is scalable and so should translate onto massively more powerful hardware in future. This will reduce repetition in processing and ultimately increase availability to end-users. In addition, access control can be improved by central database control. The results of pre-processing by PHP will be stored in a MySQL database. End-users will access this data via web technology using XML data transmissions over HTTP. Ultimately a client running on Windows and written in Visual Basic 6.0 will test the client server concept. Java programming would be preferred but is not an available as a resource.

Chapter 6, Prototype Designs

The following 12 programs have been developed as proof of concept and to demonstrate a series of key principles.

The prototype programs are available to use, complete with source code, on

http://www.mscproject.org.uk

This server, and the evidence files on it, is located in Dusseldorf, Germany.

Most of the evidence is a collection of images from Floppy Disks with FAT12 format. These were chosen for the speed of access in a demonstration environment. There is one FAT32 image of a 300MB hard disk, this is primarily to demonstrate the use of the NIST MD5 hash set. The larger, FAT32 image will not work with any of the stage 3 & 4 programs. These require FAT12 images.

All the source code can be found
on the accompanying CD
and on the server via the web page.

6.1.0 Program 1, Mounting an image file

6.1.1 Objectives:

To prove that an evidential image file can be mounted onto the Linux file system under program control.

6.1.2 Program Name

MountAFloppyImage.php

6.1.3 Description:

This program shells the Linux 'mount' command. Mount is not available to the 'Apache' user and so the Linux utility 'sudo' is used to run the mount command as if by the super user. The function returns a value to indicate the result of the shelled command. Depending on the result either the error is shown or the directory is listed.

6.1.4 Features and notes:

Although this code is not meant to be a definitive script, it contains the ***mount*** command, which is worth further notes. One of the reasons to use Linux is so good for forensic work is that it supports so many file systems when mounting an image. These include adfs, bfs, cramfs, ext, ext2, ext3, hfs, hpfs, iso9660, jfs, minix, ntfs, qnx4, reiserfs, romfs, udf, ufs, vxfs, xfs, xiafs. All these formats mount with full read/write access however as a forensic mount we do not want to allow write access. This is the other good reason for using Linux; using the –r parameter the drive is mounted as read only. We do not need to resort to special hardware; this is an operating system function. As was stated in the review of EnCase, Microsoft Windows insists on mounting any drive read/write, which is unacceptable in forensic work. As a result, Guidance Software have had to open and interpret an image file within EnCase and so write all the drivers for the recognised file formats.

Problems with the Linux Loopback device

At this stage, it appears that only 4 images can be mounted as loopback devices on the Linux 2.4 kernel based development system. Further research will need to be done on this as this system depends on the idea of mounting many file systems as evidence. One advantage of an Open Source operating system is that solutions to these low level problems can often be provided by recompiling a custom operating system.

6.1.5 Application in the real program:

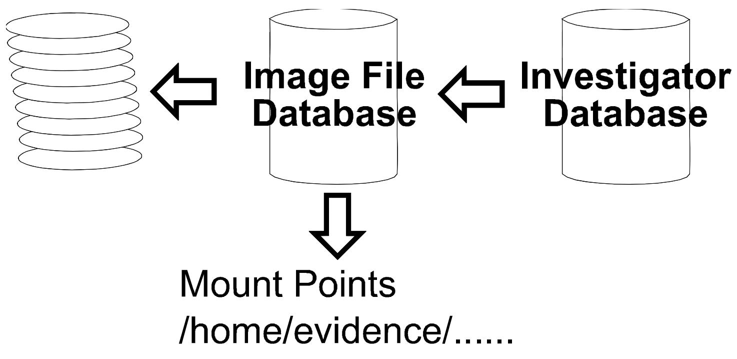

Figure 10, Mounting a File Image

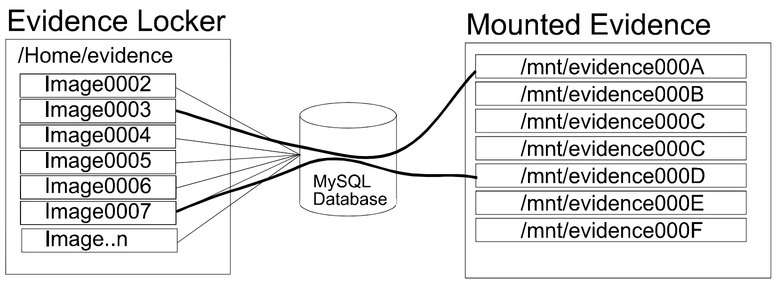

Figure 11, Mapping Images to Mount Points

A library of images, available on a massive terabyte store, would be created and administered by the highly qualified staff in charge of the server.
Disk image file names would be recorded in a MySQL database together with a corresponding MD5 hash for the entire file, description, access rights etc for the image. Access rights to individual images could then be controlled by the contents of an access level field in conjunction with a database table of user login rights.

Field	Type	Purpose	Notes
UserName	Varchar(100)	Investigator login name	Indexed
password	Varchar(32)	Contains 32 character MD5 hash of the password	Indexed
AccessLevel	Varchar(20)	Administrative access level	

Figure 12, Data Structure for Investigator Database

In this prototype program the name of the image and the mount point is hard coded. In practice each disk image would mount on a separate mount points i.e. /mnt/evidence000A, /mnt/evidence000B, /mnt/evidence000C, /mnt/evidence000D allocated by the contents of the MySQL table. Each image file would require a unique record in the MySQL database.

Field	Type	Purpose	Notes
ImageFileName	Varchar(100)	Contains the imagel file name	Indexed
md5	Varchar(32)	Contains 32 character MD5 hash for the whole image file	Indexed
Format	Varchar(20)	Format type ie FAT32, NTFS	
permissions	Varchar(12)	File Access Persmissions	
accesslevel	Integer	Administrative access level	
Mounted	Varchar(20)	If currently mounted – this is the mount point. Saves multiple mountings	

Figure 13, Data Structure for Images Table

The original images files are held in /home/evidence. The apache-php directive 'php_admin_value open_basedir' controls access to this folder from php. /home/evidence is not within the apache document tree structure and so we can be assured that access to the original images is only via php, not Apache's normal web browsing.

Although the initial proposal is that the images are located on the primary host there is no reason, other than speed, that the evidence could not be an NFS mounted drive. Additional security measures are needed to safe guard an image held off host but at this stage, this is seen as an issue beyond the bounds of this project.

<u>6.1.6 Conclusions:</u>

Images can be successfully mounted as read only. The apparent limit of four mounted drives needs to be addressed but is not expected to be a limit.

6.2.0 Program 2, Populating a MySQL database

6.2.1 Objectives:

This program traverses the newly mounted drive reading Meta file information and writing it into a MySQL table.

6.2.2 Program Name

AddFilesToMySQL.php

6.2.3 Description:

This program starts at the evidence image mount point and traverses through the entire disk structure reading all the META data associates with a file. These include MAC times, size, permissions etc. These are added to a MySQL table, *filelist*. An additional field Accesslevel is included to allow the Administrator to control access to this file by investigators. The system administrator or equivalent would then be able to set access levels on individual files within the disk image so controlling access to privileged information. Having mounted the drive image on /mnt/evidence the program will then need to populate the MySQL database. All access to the contents image is controlled by the MySQL database.

Field	Type	Purpose	Notes
Filename	Varchar(100)	Contains all the full file name	Indexed
Md5	Varchar(32)	Contains 32 character MD5 hash for file	Filled in program 3
Size	Bigint(20)	In Bytes	
Ctime	Timestamp(4)	Creation Unix Time Stamp	
Atime	Timestamp(4)	Last Access Unix Time Stamp	
Mtime	Timestamp(4)	Modified Unix Time Stamp	
UID	Mediumint(9)	File User ID	
GID	Mediumint(9)	File Group ID	
Permissions	Varchar(12)	File Access Persmissions	
Accesslevel	Integer	Administrative access level	
FileType	Varchar(12)	File type description	
KnownFile	Varchar(12)	Null/Bad/Good	Set after comparison with NIST known file database (program 4)
Category	Varchar(12)		

Figure 14, Data Structure for FileList table

This structure is appropriate for NTFS, MSDOS and Unix file systems. Additional fields may need to be added to deal with other file systems.

6.2.4 Features and Notes

Problems with certain characters in file names

The apostrophe character ‘ is allowed in file names but causes conflicts when building SQL statements like

Select * from filename where filename = “/My Documents/Nick’s Document.doc”

This in itself is not a problem but when it is used in a php program such as
$Command = “Select * from filename where filename = “/My Documents/Nick’s Document.doc””
we get problems with ‘bracketing’.

This problem will exist with the “ (double quote) and ‘ (single quote) characters. In addition, the character % has special meaning in SQL as a wildcard character. The solution is to prefix these characters with /

There can be confusion between full filenames relative to the mount point and the full filename relative to the root. The database stores actual filenames as the mount point may vary.

The sample FAT disks have hidden files. These do not retain their hidden attribute (there is no such attribute under Linux, it is controlled by a dot character at the start of a filename). Deleted file are not recovered at this stage.

6.2.5 Application in the real program:

This program would be run as part of the evidence preparation cycle before the investigators had access to the evidence.

6.2.6 Conclusions:

Directories can be successfully traversed and data pulled into a MySQL database.

6.3.0 Program 3, Calculate the database with MD5 hashes

6.3.1 Objectives:

To calculate the 32 bit MD5 Digests of all the files in the database.

6.3.2 Program Name

CalculateMD5.php

6.3.3 Description:

The program runs through all the files in the MySQL database and then looks across at the mounted file image. It then uses the built-in MD5 function within PHP to calculate the Digest. This is then stored in the MySQL table.

6.3.4 Features and Notes

This program requires only one SQL statement to select files from evidence0001.filelist but for each file, it requires the entire contents to be read from the mounted image. This is then subject to the MD5 command. This could take it a long time for a large file system! It is at this stage that the employment of batch processing becomes most important. One major time saving with this approach is that this generation of MD5s for each individual file can take place before any other processing.

6.3.5 Conclusions:

Although this could be time consuming, the MD5 digests can be calculated. As this program calls the Linux program MD5 we can inherit a high degree of confidence in the number generated.

6.4.0 Program 4, Crosscheck the calculated MD5s

6.4.1 Objectives:

To crosscheck the MD5 digests calculated in the previous program with a database of known MD5 digests.

6.4.2 Program Name

CrossCheckNIST.php

6.4.3 Description:

The program moves through the MySQL database and compares the previously calculated MD5 digest with the database of 18,000,000 digests created by NIST. Depending on the result a file KnownFile is set to either “BAD”, “GOOD” or “UNKOWN”.

6.4.4 Features and Notes

Within this design, there are processes whose aim it to reduce the workload of the investigator. The National Institute for Science and Technology (NIST, 2004) has assembled a database for nearly 18,000,000 MD5 digests for known files. These include MD5 digests for programs, libraries and images from Unix, Windows, Sun operating systems and applications. NIST are not the only collectors of MD5s but they are the most prominent. Many software authors are building their own libraries to verify their own installations.

They claim that in simple tests on a variety of PCs some 90% of files are known to be those ‘official’ files. NIST suggest the NSRL database can reduce the time needed to investigate each computer by 40% – 95% (NIST, 2004).

The NSRL database was downloaded from the NIST site in June 2004. It has been imported into a MySQL database of about 1.7GB and 18,000,000 records. In addition, for test purposes, the MySQL database has been extended to include some files identified as ‘Known Bad’. They could represent known paedophile photographic images, known copyright infringing music files or known ‘hacked’ software.

Specifically NIST describe the June 2004 data set as

4,644,674	non-English files
2,513,772	Operating System Files
7,545,675	Applications Files
3,205,843	Images

Data file structures

There are four associated NSRL databases held in MySQL formats

NSRLFile – Known Files – 17,909,964 records			
Field	Type	Purpose	Notes
MD5	Varchar(32)	Contains 32 character MD5 hash for file	Indexed
FileName	Varchar(40)	Contains file name	Indexed
FileSize	Bigint(20)	In Bytes	
ProductCode	Varchar(11)	The Code in the NSRLProd table	
OpSystemCode	Varchar(11)	The code in the NSRLOS table	
SpecialCode	Varchar(10)	Unknown	
Category	Varchar(10)	"Good","Bad"	

Figure 15, Data Structure for NSRLFiles table

NSRLOS – Operating Systems – 114 records			
Field	Type	Purpose	Notes
OSCode	Varchar(32)	Contains 32 character MD5 hash for file	Indexed
Description	Varchar(40)	Contains file name	
Version	Bigint(20)	In Bytes	
Manufacturer	Varchar(11)	The Code in the NSRLProd table	

Figure 16, Data Structure for NSRLOS table

NSRLProd – Products – 4,873 records			
Field	Type	Purpose	Notes
Code	BigInt(20)	Unique ID	
Name	Varchar(30)	Contains file name	
Version	Varchar(30)	In Bytes	
OpSystem	Varchar(30)	The Code in the NSRLOS table	
Mfg	Varchar(30)	Manufacturer Code	
Language	Varchar(30)	Human language	
Type	Varchar(30)		

Figure 17, Data Structure for NSRLProd table

NSRLMfg – Manufacturers – 533 records			
Field	Type	Purpose	Notes
MfgCode	Varchar(10)	Unique ID	
Name	Varchar(30)	Manufacterer's Name	

Figure 18, Data Structure for NSRLMfg table

It is surprising that with nearly 18,000,000 files there are still about 20 files on the test floppy images that are not in the NIST data set. These files are from a very old version of FoxBase for DOS.
The program runs through the records in evidence0001.filelist and tries to match them in NIST.NISTFiles. If it succeeds, files with NSRLFile.Category set as “Bad” are highlighted in Red, with NSRLFile.Category set as “” or “Good” or Null is highlighted in Green. All other files are left white. The evidence0001.KnownFile field is set appropriately.

In this prototype, set-up programs 2,3 and 4 are run as separate stages but would be run as batch by the administrative installer.

6.4.5 Conclusions:

Cross checking the MD5 digest is a very worthwhile procedure. Potentially incriminating evidence can be identified with a high degree of certainty without revealing the actual evidence to the investigator. This could be very useful in sensitive investigations when investigators need to be shielded from the actual evidence as with some aspects of national security or paedophile photograph investigations.

6.5.0 Program 5, An XML Server

6.5.1 Objectives:

To obtain an XML format delivery of the contents of the evidential database.

6.5.2 Program Name

XMLServer.xml

6.5.3 Description:

This is an example of the availability of Open Source software that could be included in a larger package. The SQL2XML package was obtained from the PHP Extension and Application Repository at pear.php.net. This is typical of the Open Source ethos in that individuals are invited to contribute their developed software to a source of reusable components for PHP. The main issue here is that of licence not of technical development. All contributions to PEAR are subject to acceptance that the licensing conforms to PHP, Apache, LGPL and BSD style licences.

This PHP program accepts an SQL statement as a parameter in a URL and returns the dataset as well formed XML.

http://www.mscproject.org.uk?Query=Select * fromevidence0001.filelist limit 5

Although the program is Open Source, the contents will not be discussed in this project. The program simply serves provide a jump into XML work with a known working base. This function is fulfilled by programs in Program 10.

As discussed in Chapter 3, XML is rapidly becoming the established means of data transmission over the Internet.

6.6.0 Program 6, A PHP XML Client

6.6.1 Objectives:

To establish a connection between a client php program and the previous program, the XML server.

6.6.2 Program Name

xml-reader.php

6.6.3 Description:

This program, written in PHP and running on the secure server takes it's input from a URL request sent to SQL2XML. The returned data is parsed and displayed in an HTML table.

6.7.0 Program 7, A PHP XML Client with user input

6.7.1 Objectives:

To further program 6 with the addition of user input

6.7.2 Program Name

xml-post.html

6.7.3 Description:

This program allows the user to enter an appropriate SQL statement in the text box on the screen. It is not apparent to the user but the script calls xml-reader.php with a POSTed variable Query. This is passed onto XMLServer.xml as a GET string. The resulting text is then parsed and the result shown as a table. This is a very simple function with access limited to one database with three tables. There is no validation of user input.

6.8.0 Program 8, Binary access to a logical file

6.8.1 Objectives:

To produce a hex dump of a file accessed as a logical file.

6.8.2 Program Name

HexView.php

6.8.3 Description

This program simply opens a file on the mounted image file and displays the contents in hex.

6.8.4 Features and Notes

The program uses the PHP function fopen to open "/fileba~1.txt", we should note from this program that PHP does not read characters after the EOF marker. Later programs can reveal this data.

6.8.5 Conclusions

This method is not suitable when searching slack space, unused clusters or clusters marked as bad.

6.9.0 Program 9, Reading the MBR of a FAT12 Image

6.9.1 Objectives:

To access the evidential image file directly at the binary level and extract useful information.

6.9.2 Program Name

ReadImageFileFAT12.php

6.9.3 Description

This program opens the evidential image file directly and reads in the Master Boot Record (MBR) data. The MBR is a suitable small limited entity as it is a specific format and contains useful information.

6.9.4 Features and Notes

This program is specifically written for FAT file systems and would need to be tailored for each format type. Although Linux deals with many file type when disks are mounted of the file system as logical extensions, this is of no use for our access to the binary data. The information extraction that the mount command performs would have to be mimicked in this system.

6.9.5 Conclusions

Drivers (or more specifically information layout readers) would need to be written for each file format expected.

6.10.0 Program 10, Reading FAT12 and a root directory

6.10.1 Objectives

To read in the File Allocation Table (FAT) and the directory from an image of a floppy disk.

6.10.2 Program Name

HexViewFAT12Viewer.php

6.10.3 Description

The program locates the FAT within the image file, extracts information and then inserts it in the evidence0001.FAT table. It then proceeds to interpret the root directory.

6.10.4 Features and Notes

We know that the image analysed is going to be FAT12 so no deduction of the image format is done. The FAT is traversed, extracted, interpreted and inserted into a MySQL table – FAT. FAT12 has an intricate structure and so is a good test of the extraction abilities of PHP. The cluster is located within the image and a MD5 digest is calculated and stores in evidence0001.FAT. For test purposes a random number between 1 and 100 is generated and stored as an access level number. This is entirely for test purposes and would be set by and administrator in a real system.

FAT table			
Field	Type	Purpose	Notes
ClusterNo	SmallInt(6)	Contains the cluster number	Indexed
Contents	Varchar(6)	Contains the next cluster number used by the file, bad marker, EOF marker etc.	
MD5	VarChar(32)	MD5 Digest for the Cluster	
AccessLevel	Interger	Access Control Value	

Figure 19, Data Structure of evidence0001.FAT table

Then the root directory is analysed. Long File Names are interpreted, deleted files are shown and file attributes are interpreted. Using the array built in the first stage, the program traces through the appropriate entries in the FAT terminating at the EOF entry for the file. The program calculates the number of bytes in the file and hence the remaining bytes in slack space.

This image of a floppy disk contains 2880 clusters. Subsequently 2880 records are created in evidence0001.FAT, each one is recorded in evidence0001.FAT. Each record is 45 bytes and so the record for a single floppy image would result in a FAT table of 500k. If this was extended to a 128GB FAT image, the figures become much larger. A sample 128GB disk, from a typical machine, contains two FAT volumes.

Size	Number of Files	Filelist table file size	Number of Clusters	FAT table File size
60GB	124,000	11MB	122,000,000	5.5GB!

Figure 20, Expected MySQL Tables sizes with a large volume

Although no research was done into other file system structures we believe that NTFS uses a very compact representation of just 1 **bit** per cluster in its storage of allocated cluster information and then storing only used cluster information in with the file meta information.

Further Problems with large image files

As a standard compilation, PHP uses a 32 bit long integer as a file pointer when opening any file for input. 2^{32} = 4,294,967,296. So the biggest file PHP can normally open with fopen(is about 4GB. This would be a major problem with forensic work as we have already argued one of the reasons for a remote investigative system is that evidential media will increase in size. 4GB hard disk drives were standard in 1999. As PHP is open source we are able to compile a custom installation setting CFLAGS=”-D_FILE_OFFSET_BITS=64”. This sets the pointer to use a 64 bit number or ‘long long’. This increases the file pointer to $2^{64} = 18 \times 10^{18}$.

6.10.4 Conclusions

This suggests that on large evidential volumes there could be a 10% storage overhead when using MySQL databases.

6.11.0 Program 11, A Visual Basic, XML Client, File List

6.11.1 Objectives

To create a Visual Basic Client to request and parse an XML dataset.

6.10.2 Program Name

VB6XML.vb

6.10.3 Description

This program extends the XML Client/server idea still further. Written in Visual Basic 6 this program runs as an installed client on a PC running Windows. Its task is simple, to request data from the PEAR XML server. The data is called in, parsed and displayed on a simple grid. This proves, in a very simple way, that XML data can be requested and parsed in a VB program.

6.12.0 Program 12, A Visual Basic, XML Client, Hexadecimal Viewer

6.11.1 Objectives

To allow an investigator to move through the evidential image held on the server.

6.11.2 Program Name

On the Client - XMLCluster.bas, on the Server - XMLSector.xml & XMLFat.xml

6.11.3 Description

This is the culmination of this research project and it should be considered as a small suit of programs.

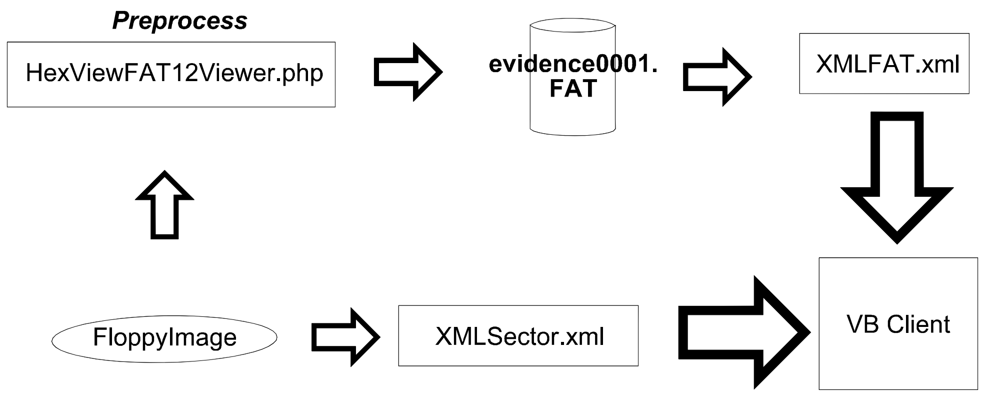

Figure 21, Data Flow through to the VB Client

HexViewFAT12.php	(Program 10) Pre-processes the data creating evidence0001.FAT containing cluster and MD5 data.
VB XML Client	the user application to display Hex data of the Floppy Disk Image
XMLSector.xml	a server application to provide the VB client with information from the sectors of the Floppy Disk Image
XMLFAT.xml	a server application to provide the VB client with information from the FAT table containing information about the structure of the FloppyDisk Image

6.11.4 Features and Notes

This VB client does several things.

- On loading, the form requests data from the FAT table (filled in program 10).

- As a noise reduction feature, and hot spot highlighting, specific parts of the FAT data are shown. This data has been filtered to list only the data of primary importance to the forensic investigator.

    - It lists any cluster marked as a file EOF, this is where 'slack space' occurs. This version simply shows any EOF cluster but a more refined version of program 10 would calculate the actual EOF position and only highlight clusters where the slack space contained some data i.e. not 0xF6.

    - It lists clusters marked as BAD. Importantly it shows not all BAD clusters but only those with an MD5 hash that is not 'f49ed788acc2753e5a1736808dcdd138'. This is the MD5 for 512 bytes of character 0xF6, which is used to fill all the bytes in a sector when the original floppy is formatted.

   - It displays cluster number for those clusters that are marked as unused but do not have an MD5 hash of ‘f49ed788acc2753e5a1736808dcdd138’, i.e. those cluster that DO have data. These could be the remains of deleted files and may not be retrievable.

- Any cluster can be viewed by editing the cluster number box at the top of the form and then pressing GO. The cluster number can be increased and decrease by the up/down tool.

- The user can set an access level. Any cluster having had its access level set in program 10 at a lower level cannot be viewed, displaying an appropriate message instead.

The program has some rudimentary in error detection built-in (see chapter 7.5). As part of the cluster status information retrieved from evidence0001.FAT, the MD5 hash is retrieved. This was created when program 10 was run. When the actual data is retrieved from the disk image it is transmitted to the client via XML but a new MD5 checksum is calculated in the client software. The MD5 hash retrieved from the server is compared with the newly calculated MD5 hash. The two are displayed at the top of the form together with a tick, or cross, appropriate to success or failure.

*The Visual Basic MD5 function comes from CryptoSys Hash http://cryptosys.net/hash.html In use, frequent errors can be found with discrepancies between source MD5s and Client calculated MD5s for the same cluster data. As these errors are consistent, they are most likely to be the result of bugs in the client library not actual transmission errors.

The client program has many limitations but these are only imposed by the timed time available for development work. One important limitation is that it only allows access to clusters i.e. sectors used for storing data not raw sectors i.e. those used to store root directory and FAT information.

Potential Communications Workloads

Initial attempts at extending the program to include the ability to track through the clusters allocated to a specific file highlighted a specific problem.

Imagine a process thus:

1. A user clicks on a file name on the client program.
2. A request for the file metadata is sent to the XML FileList server
3. The response includes the number of the first cluster used by the file
4. The client requests the cluster details from the XML FAT server
5. The cluster data is sent from the server
6. The client looks the data, if not EOF return to 4
7. Display the FAT cluster sequence on the client

As a ordinary file could be 10MB, i.e. 20 clusters, this could take some time as each loop requires an http request and response.

In this case, it would be better to use the pre-processing approach to gather and transmit the cluster information with the file name when the initial request for the starting cluster is made.

1. A user clicks on a file name on the client program.
2. A request for the file metadata is sent to the XML FileList server
3. The response includes all clusters used by the file
4. End of process

Chapter 7, Testing

7.1 Introduction

In this chapter, we consider what would be needed to test the software developed so far and the whole system as it develops further. In looking for a test methodology, it becomes clear that although standards and testing methodologies exist for many things they are not so developed for digital forensics. This is presumably because of the short history of digital forensics. It is sensible then, to take lessons from what little testing currently exists and adapt them for our purposes.

In the American Courts, digital evidence is subject to a “Daubert Hearing” to decide weather is admissible in court (Carrier, Open Source Digital Forensics, p3, 2003). It is subjected to four criteria

- **Testing**: Can and has the procedure been tested?
- **Error Rate**: Is there a known error rate of the procedure?
  Are there false negatives or false positives?
  Are there tool implementation errors?
- **Publication**: Has the procedure been published and subject to peer review?
- **Acceptance**: Is the procedure generally accepted in the relevant scientific community?

Publication and Acceptance are matters for the wider forensic community. Within this project, the three items within Error Rate is most important.

On their web site MySQL state

> "**Reliability and Performance**. MySQL AB provides early versions of all its database server software to the community to allow for several months of 'battle testing' by the open source community before it deems them ready for production use"

this is typical of the approach of most of the Open Source community and is also true of PHP and Apache. MySQL have a business partner arrangement with several vendors to provide certified server performance. In this way, this project's error checking of the base software is inherited from the original authors.

A greater concern within this project is the possibility of tool implementation errors. These prototype programs were written to analysis a FAT12 DOS format floppy disk. Microsoft publishes this specification and so it can be taken as authoritative. The program results have been tested by cross checking against the results of another known program considered to be an authority, in this case Norton Disk Editor. This may not be true of all the other disk formats that could be encountered. There is no official detailed specification published for NTFS, for example. If this project were extended to the analysis of this format, information would have to be sought from non-authoritative sources. The Linux-NTFS project is one such (Linux-NTFS Project, 2004). Just three individuals currently run the project. If they get it wrong, we get it wrong. This is true, however, for the likes of Guidance Software as well, as they have chosen to assure a read only state by 'importing' images and opening them within their application, they must be going through the same process. Again, it is appropriate to accept the testing of these formats by the groups involved in their investigation but acknowledging when they do not test to high enough standard. Within an extended version of this project, comparative testing would need to be done to create a level of assurance in the interpretation of these formats in the project programming.

7.2 NIST and CFTT

The National Institute of Standards and Technology (NIST) has a dedicated group for Forensic Tool Testing – the Computer Forensics Tool Testing Program (CFTT). They are currently developing tests and test methodologies specifically intended for testing digital forensic software and hardware. It has to be said that as digital forensics testing is only in its infancy CFTT has currently only tested three or four classes of product, Disk Imaging software is the nearest and most appropriate for our purposes. It is not too near a match but the reports give and insight into their thinking on the test procedure.

CFTT analysed the 'dd' program supplied with Red Hat Linux 7.1 version 4.0.36 (NIST, August 2002). This is of interest because in the development system used in this project the 'dd' program was used to capture images. In this particular case 'dd' version 4.5.3 was used.

7.3 CFTT testing Disk Imaging Software

In General Test Methodologies for Computer Forensic Tools v 1.9 (NIST, 2001) CFTT state that their attempts to formulate a testing methodology is complicated by a lack of standards or specification of what forensic tools should actually do and that their developed methodology is based on experience with other standards and type of software and hardware. They say that their work is based on well-recognised standards such as ISO/IEC Guide 2, Standardization and Related Activities – General Vocabulary.

CTTS's testing methodology is a detailed 39-page document. The key points raised are:

Testing transparency and peer review of procedures. CFTT state that the purpose of their testing specification document is to enable others to rigorously scrutinise their method and reproduce the tests as validation. Subsequently CFTT proved a painstakingly detailed specification that covers:

1. 52 cases of varying scenarios
    a. IDE and SCSI source and destination drives,
    b. With and without hardware errors on the source disk.
    c. With and without logical errors in the disk structure
    d. With identical and dissimilar sizes of source and destination disks

    CFTT have tried to identify situations that could cause errors
2. They then define in very specific terms
    a. The host computer.
    b. The drives they used (Harddisks, Jaz disks, Floppy boot disk and Linux boot disk.
    c. The formats and state of information setup on the source drives.
3. They then define their procedure in detail their procedure for
    a. setting up a source hard disk:
    b. setting up a destination hard disk
4. They list their scripts
5. They specify how they analyse the resulting log files

CFTT's test methodology is an impressive document. In Brian Carrier's "Open Source Digital Forensic Tools" he suggests that

"… the testing requirements are likely to be stricter for digital forensic analysis tools than the original application or operating system tests."

Indeed, it is unlikely that the author of dd, Rubin, MacKenzie and Kemp would have tested their own creation as thoroughly as it was by CFTT.

7.4 A Test Methodology for the prototype programs

The prototype suit generated from this project was always intended to cover a wide range of issues to prove that various components can be built. The main reason for taking a prototype approach was that as there were no existing software solutions of this type available and so there were no established approaches. In essence we need to assure ourselves that the data represented by the image is interpreted, stored, retrieved, transmitted and finally displayed by the client software is as free from errors as possible, or if errors occur they are possibly corrected automatically or at least detected and reported. However drawing analogues from CFTT's work we can see that a few specific parts of the prototype suit stand out for testing.

1. The extraction of FAT and Directory information in program 10.
2. Reading and writing information to the MySQL database
3. Verify the risk of errors when generating XML from database reads and transmits across the Internet because of requests from clients.
4. The reception of data in XML format and it's display as hexadecimal

7.4.1 Test Task 1

Description: Verify that the results for the script generate accurate information reflecting the true contents of the FAT on the image.
Method: Extract data and compare the extracted information with an accepted authoritative package possibly Norton Disk Editor.

7.4.2 Test Task 2

Description: Verify that data can be written to and read from the MySQL with a high degree of certainty.
Method: Write one php script to exhaustively generate random data, add it to the database. Use another script to exhaustively read it back and verify it against the original data.
Notes: Consult MySQL web site for data on reliability.

7.4.3 Test Task 3

Description: Verify the risk of transmission errors while across the Internet (TCP/IP link).
Method: Setup 2 machines, one as server, one as client. Transmit a series of control data sets, because the source data can be calculated the client can be set to expect a predetermined sequence of data.
Notes: Internet links can fail in 2 ways. There could be data corruption or there could be data dropout where data sent never arrives.

7.4.4 Test Task 4

Description: Verify the correct reception and interpretation of the data in the client.
Method: Create a test set of XML data on the server and repeatedly trigger its reception by the client.

7.4.5 Test Task 5:

Description: Whole System Test.
Method: Store a disk image on the server and a copy on the client. Compare the resulting data received on the client against the local copy of the evidence image. A variety of images have been loaded on the server. The results shown by the VB client hexadecimal viewer were compared with the local results of the original floppy disks viewed through Norton Disk Edit. Within the time constraint, no discrepancies were found. Further testing would need to be automated so that a much larger testing sample could be used.

Although there are clearly stages to the data transfer task 5 tests all five. It seems to be a labour saving decision to run validation Task 5 and then only errors are detected do we need to look further into where they occur.

7.5 A self-testing, self-assured, system

Perhaps a better approach would be to build-in data integrity within the programming in this project. In the final program, 12, the hex viewer uses a local MD5 calculation to validate its own data.

During program 10, as the program moves through the File Allocation Table, it is calculating an MD5 checksum for each cluster. This is stored in the evidence0001.FAT table. When program 12, the VB client software, requests a cluster of data (via an XML call) a parallel call is made to the FAT table and the previously calculated MD5 (calculated by PHP) is also retrieved. When the data arrives at the client, a small Visual Basic API calculates a fresh MD5. It is important that this is a different routine than first calculated the MD5 stored in the FAT table. The two MD5s are shown at the top of the form with a tick or a cross to show a match or otherwise. This display is for prototype purposes only, in a production version, this error checking and retesting would be less conspicuous. This dynamic provides validation through the whole chain of evidence transfer.

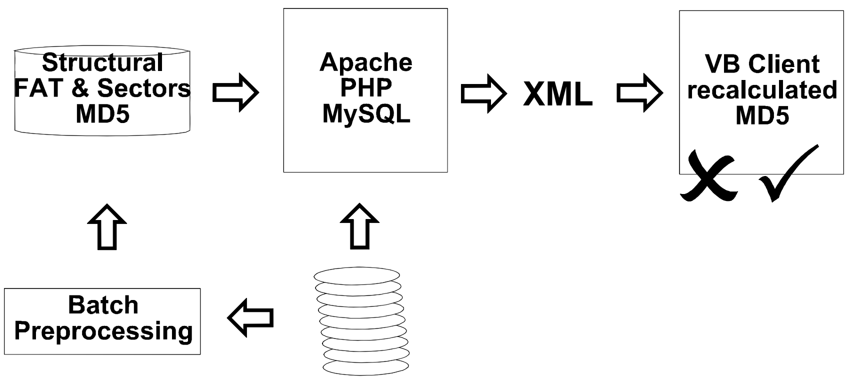

Figure 22, MD5 feedback recalculation

During testing, it became apparent that there were bugs in the MD5 library obtained from Cryptosys Hash. The MD5 created on the server was cross checked with WinMD5 v2.01 by eolson@mit.edu The digest calculated by PHP-MD5 and WinMD5 agreed, the MD5 calculated by Cryptosys Hash was different on the same data. It is outside the scope of this project to write an MD5 digest generation program for Visual Basic. This could be addressed elsewhere.

7.6 Securing XML

XML has no inherent security or data validation. This would be very desirable in this application. The SSL encryption built into HTTP provides a secure transmission. As the XML server is driven by PHP it seems sensible to use the session security built-in to PHP to control access to the supply of XML data.

7.7 Packages of evidence

The concept of a package of evidence can be developed from the ability to add an extra record at the end of each data transmission with an MD5 digest of the data within the transmitted package.

XML Record 1	Record of evidential data
XML Record 2	Record of evidential data
XML Record 3	Record of evidential data
…	
XML Record n	Last Record of evidential data
XML Record n + 1	MD5 Digest of previous recorded data

Figure 23, A Package of evidence

7.8 Conclusions

Forensic software probably needs higher testing standards than the original software. An Open testing system would be more appropriate because of the use of the results. There are no established testing standards for forensic software. Systems can inherit errors and mistakes from their components. Components need to be tested as well as a whole. It would be better to develop a system with inherent testing and checking from end to end.

Chapter 8, Evaluation

8.1 Introduction

The project can be evaluated by assessing the way in which the project satisfied the design objectives set out in 1.4 and 5.4.
They were:

1. To enable control a library of binary images of evidential disks
2. To control access to the images
3. To pre-process the disk files to reduce 'investigative noise'
4. To analyse and make sense of a disk at a binary level and display the results.
5. To convey this information to the user in the form of a simple hex viewer like Norton Disk Editor

8.2 Objective 1

To enable control a library of binary images of evidential disks

In a final working program, the client software would have to perform this function. To enable multi-user access, each image would need a separate mount point and the creation of a database with MySQL. This project allows the user to mount an image of their choice from the web site but does not all multiple users to view different images without conflict.

8.3 Objective 2

To control access to the images

Program 1 allows the investigator to choose which image to view. It does not validate against an access level in a database of Investigator's information. In this respect this was a failure but it is felt that is was only a failure because of the time constraint. It was more important to develop the project to add features to the final VB Client program rather than implement a feature in a prototype.

8.4 Objective 3

To pre-process the disk files to reduce 'investigative noise'.

This is was very successful. MD5 Digests were created and checked against the NIST database. Performing this function against the 'Small Hard Disk Image' supports NIST's assertion that 90% of files can be eliminated this way. This is a key method in reducing the work load of the investigator and so time spent.

8.5 Objective 4

To analyse and make sense of a disk at a binary level and display the results

Within the limitations of the FAT12 disk structure, this was very successful. The information in the FAT was retrieved and analysed, it was inserted into a MySQL database and was retrieved by the VB Client. Most of the Master Boot Record information was decoded until it was seen that anything could be read.

8.6 Objective 5

To convey this information to the user in the form of a simple hex viewer like Norton Disk Editor

The Visual Basic Hex Viewer client was very successful. The user can scroll around the evidence viewing the contents of clusters. Interesting clusters are highlighted on the left; a result of pre-processing on the server. Investigator access control can be demonstrated by changing th e access level value.

8.7 Conclusions

Within the crucial time limit of 16 weeks, the project has been a qualified success. The VB Client was written and forensic data was passed from a server, located in Dusseldorf, to the client wherever it was installed. On a 64K ISDN line this was delivered at a tolerable speed, on the very fast link available at Glamorgan University there was only a marginal difference in response when compared with a local task. It has established that an XML client server forensic program can be written. All the principle concepts were addressed and successfully completed. The project would have generated more valuable results had the server program been written in C and the client been written in Java but neither programming skills were available. This limit was always understood by the programming skills available.

Chapter 9, Possibilities for future development

9.1 Introduction

As this project proposes the application of a client/server approach in a field in to it seems to not have been applied before, there are a large number of areas that need further investigation. We feel this justifies a chapter in its own right.

9.2 Large disks

Most of the development work for this project was done on a 1.44MB image of a Floppy Disk with about 30 files. It was felt that, although small, a floppy disk represented most of the problems that would be encountered in the real application of this approach. In fact, there is a 300MB image of a Windows 98 system on the server. Brief testing of this was very promising. An image of this size uses FAT32 and having spent considerable time researching FAT12 the time constraint on this project did not allow a FAT32 analysis tool to be developed. When the demonstration program is run it takes some time to process the 3000 files found on the Small Hard Disk Image, this is mainly the transmission of the audit trail and presentation processing of the browser.

9.3 Further investigation of the use of MD5 databases

When testing the larger 300MB image nearly 90% of files were recognised by the NIST database. This reduced the potential search area to just 30MB. When combined with the elimination of blank unused clusters the quantity of data considered during a search could be massively reduced.

9.4 An on-line XML MD5 lookup database

Expanding the previous item, if an MD5 database was available as an on-line XML data source there is a possibility that a bootable CD like Knoppix for Linux or a bootable CD for DOS could be developed to scan host media and report back on suspect files found without writing to the media.

9.5 Data Caching

Having retrieved data, either by cluster or by file, it seems sensible to cache it in the client in much the same way as Internet Explorer caches its data. Although the link is surprisingly fast on a 100MHz network caching would benefit slower ADSL connections. The local cache would probably have to be encrypted with a key linked to the current session; known only to the software and that would expire after a pre-determined time. This would ensure the security, confidentiality and integrity of the data when stored locally.

9.6 Text Searching

Following the pre-processing approach used in the MD5 calculation, NIST database cross check and FAT analysis it needs a search function on the server side. In fact its not practical to do this on the client side bearing in mind the data to be searched would have to be transmitted to the client in order that the client could do the search. The power of the server would make searches far faster. The pre-processing of data can make huge savings on time on searches. By eliminating all known file space and all unused file space that is actually blank a massive search task presented by a very large hard disk could shrink dramatically. If this was combined with the work by Dr Richard's in New Orleans even more dramatic search times could be expected.

9.7 Investigator and Case Handling

Although there is a Case Management table within evidence0001 it is only used for the mount point for images within the file system. This would be the heart of a Computer Supported Co-operative Working environment. The records in both the tables evidence0001.filelist and evidence0001.FAT have a access level field. It is intended to set an 'access level' for each file and cluster on the evidential image. If each investigator was also allocated an 'access level' selective access to the contents of the image could be controlled. Investigators could access those parts of the image relevant to their investigation while restricting access to others for reasons of confidentiality, security or privacy.

9.8 To Develop a Formal Open Source definition for the XML transmission of forensic data.

In anticipation of developments in forensic software, there may be a need for a standard for data transmission. The Cyber Tools On-Line Search for Evidence project (CTOSE, 2004) are funded by the European Union. They have created a set of XML DTDs allowing the transmission of packages of evidence. Unfortunately, these were not available to this project at the time of writing. If such a standard was accepted it could form a basis for future open forensic tools and an open route for data in propriety software from manufacturers like Guidance.

9.9 Greater pre-processing of evidence image files

Bearing in mind the anticipated increase in media size and quantity needing forensic investigation the server side pre-processing could be greatly enhanced to improve noise filtering and the focus on hotspots of evidence.

Data Extraction. The investigation of many known data files like .pst, for Microsoft Outlook, and .dta, Sage accounts files, require the original mother program to be run to investigate the data held in the files. Modules could be written, some already have been, to translate the contents of these binary coded files into text that would allow them to be searched during the pre-processing stage.

Deleted Files Although the Hex Viewer can see the contents of deleted files, this project does not attempt to recover them. Brian Carrier's Autopsy utilises Weite Venema's Sleuthkit to recover similar information from Unix systems. It would be sensible for sub programs, perhaps written in C, to recover deleted files to be used to do the same type of work.

Virus and Trojan scanning would need to be done at this stage. This does not represent a problem because a standard anti-virus package could be used to simply report on the presence of virus signatures without fixing them.

9.10 Reducing data transmission speed problems

As the transmission overhead is quite large with XML (see Chapter 4) the quantity of data need to transmit pictorial information for a gallery view on the client could be reduced by using software such as Netpbm to create thumbnails. These thumbnail images could be transmitted rather than the originals. This could mean reducing the data transmitted from a 1MB picture to 35k.

9.11 Developing Collaborative work tools

To create a full workplace environment the forensic functions could be augmented by the development of more general communication tools like scratch pads where investigators could share thoughts and clues.

9.12 Conclusions

This project may generate more questions than it answers but having established that the client/server design can work and there are potential advantages that single PCs or even LANs cannot deliver the questions are probably worth answering.

Chapter 10, Conclusions

In chapters two and three we established that digital forensics is a young and rapidly developing subject. In such circumstances, predicting the future, even in the short term, would be helpful but likely to fail.

This project asserted that there is a serious obstacle confronting the investigator in the near future. Because of the uneven development of disk storage and disk transfer speeds, a bottleneck has occurred that will cause the forensic investigator great problems. Disk capacities have increased but transfer rates have not kept pace with this change in capacity. It will still take many hours to extract data from the original evidence but having extracted it further analysis could be greatly improved by using mainframe or supercomputer facilities rather than PCs. This could provide multi-user distributed access and processing. Some may call this groupware, a collaborative system or CSCW. Surprising we found almost no research in this subject area of forensic tools. This project explored the possibility of a forensic program in the client server idiom. A full system was always well beyond the scope of this project but we hoped to establish some practical demonstrations and from there suggest further research and development.

We have reviewed what little relevant literature was available. Key tasks were identified by drawing from the works of Kruse and Casey and the main features of EnCase, Autopsy, XML Forensic Tool were brought together to generate the features that should be included when designing a forensic tool in the future.

We considered the nature of connections over wide area networks, namely the Internet, and saw that a client-server configuration is the best solution to overcome its weaknesses. In choosing a data transfer method to deliver data to the user, we considered HTML and the web browser and acknowledged that it is too limited to provide a sophisticated user interface. Enhancement of the HTML, browser combination with client scripting would generate security concerns within the potential user groups. Ideally, Java seemed to provide all the features we need. Given this, XML is the current choice as an open protocol to transfer data.

The programs were written as prototypes allowing the project to develop organically. A relatively plain Pentium 2.0GHz processor running Redhat Linux 9.1 was used but all the system software was chosen to be scalable so the result would be valid on a variety of host systems.

The results of pre-processing by PHP were stored in a MySQL database. End-users initially had access via web technology using HTML and later by XML. Java programming would have been preferred but was not an available as a resource so ultimately a client running on Windows was written in Visual Basic 6.0 to test the client server concept.

We acknowledged that forensic software probably needs higher testing standards than the original software. An Open testing system was thought more appropriate because of the use of the results in litigation. As the subject is so new, it was no surprise when we found that are no established testing standards for forensic software. We acknowledged that systems are at risk of inheriting errors and mistakes from their components. Components need to be tested as well as a whole. It would be better to develop a system with inherent testing and checking from end to end. Because of this, the final program was amended to include a check based on an MD5 digest.

Within the crucial time limit of 16 weeks, the project was a qualified success. The VB Client was written, an XML server was written and forensic data was passed from a server, located in Dusseldorf, to the client wherever it was installed. On a 64K ISDN line this was delivered at a tolerable speed, on the very fast link available at Glamorgan University there was only a marginal difference in response when compared with a local task.

This project has established that an XML client server forensic program can be written. All the principle concepts were addressed and successfully completed. The project would have generated more valuable results had the server program been written in C and the client been written in Java but neither programming skills were available. This limit was always understood.

When pre-processing, data filtering was particularly successful when the 300MB disk image was processed using MD5 digests. The idea of processing on a supercomputer like Virgina Tech's Terascale computer would mean that searches were truly interactive. To justify such a facility, wide area distribution of results would be essential.

As this project was concerned with base work, it was expected that it might generate more questions than answers. Having established that the client/server design can work and there are potential advantages that single PCs or even LANs cannot deliver the questions are probably worth answering.

The work in this project is not meant to solve a problem that exists today but it to solve a problem that will soon exist.

Perhaps we might listen to the words of Alan C. Kay of Apple

> "Look, the best way to predict the future is to invent it." (Kay, *1989).*

Bibliography and References

Association of Chief Police Officers (ACPO), 2004, **Good Practice Guide for Computer Based Electronic Evidence**, http://www.nhtcu.org/ACPO%20Guide%20v3.0.pdf, accessed 5th July 2004

ACPO, 2004, **Revised Guidance for the Control of Paedophile Images**, http://www.nhtcu.org/Revised%20Guidance.pdf, accessed on line 23rd September 2004

Bradley, N., 2000, **The XML companion**, Pearson Education, Harlow, UK

Britz, M. T., 2004, **Computer Forensics and Cyber Crime**, Pearson Prentice Hall, London

Bratby, L., 2004, **Hi-Tech Moves**, **Police Review**, p23-24,

Carrier, B. 2004, **Open Source Forensics**, http://www.opensourceforeniscs.org/tools/unix.html, A library listing of Open Source Forensic Tools for Unix, Accessed 28th Aug 2004

Carrier, B. 2004, **Digital Forensic Tool Testing Images**, http://dftt.sourceforge.net/, accessed 26th August 2004. A collection of 'medium sized' disk images for testing purposes.

Carrier, B. 2003, **Open Source Digital Forensic Tools**, available on-line at http://www.atstake.com/research/reports/atstake_opensource_forensics.pdf accessed 26th August 2004

Casey, E., 2004, **Digital Evidence and Computer Crime**, 2nd Ed., Academic Press, London

Casey, E., 2002, **Handbook of Computer Crime Investigation**, Academic Press, London

Choi, W. et al, 2003, **Beginning PHP**, John Wiley & Sons

Computer History Museum, 2004, on-line at http://www.computerhistorymuseum.org/, accessed 14th September 2004

Crabtree, A., 2003, **Designing Collaborative Systems**, Springer, London

Cryptosys Hash, **Cryptographic APIs for VB, VC etc**, on-line at http://www.cryptosys.net/

CTOSE, 2004, **Cyber Tools On-Line Search for Evidence**, on-line at http://www.ctose.org, accessed 17th September 2004

Dell, 2004, **Dell Dimension 2400**, on-line at http://www.dell.co.uk, accessed 26th September 2004

Eagleston, B and Ridley, M, 2001, **Web Database Systems**, McGraw Hill, London

Gulker, C., 2001, **The Kevin Mitnick/Tsutomu Shimomura affair,** on-line at http://gulker.com/ra/hack/ accessed 17th September

H M Government, 2004, **Interoperability**, on-line at http://e-governmant.cabinateoffice.gov.uk/Briefings/BriefingsArticle/fs/en?CONTENT_ID=4000204&chk=jPx9fy, accessed 6th September 2004

Home Office, 1984, **Police & Criminal Evidence Act 1984**, On-line Halsbury's, http://www.butterworths.co.uk/halsbury/index.htm, accessed 23rd September 2004

Home Office , **Criminal Justice & Police Act 2001**, On-line, http://www.butterworths.co.uk/halsbury/index.htm, accessed 23rd September 2004

(ICS) Intelligent Computer Solutions, 2004, **Flastbock** on-line at http://www.icsforensic.com/, accessed 23rd Sept 2004

Kovacich, G. and Boni, W., 2000,
High-Technology-Crime Investigator's Handbook, Butterworth Heinemann, Boston

Kay, A., 1989, Predicting the Future (first published in ***Stanford Engineering,*** *Volume 1, Number 1, Autumn 1989, pg 1-6)*, on-line at http://www.ecotopia.com/webpress/futures.htm, accessed 16th September 2004

Kruse, W. and Heiser, J., 2002, **Computer Forensics,**
Incident Response Essentials, Addison Wesley, Boston

Koukouras, Y. & Vlastos, E., 2003, **Forensic Tool with XML**, on-line at http://sourceforge.net/projects/ftxml, accessed 15th August 2004

Lafon, M., 1999, **Computer Supported Co-operative Work**, Wiley, New York

Linux-NTFS Project, **NTFS Documentation**, on-line at http://linux-ntfs.sourceforge.net/ntfs/

Littlejohn Shinder, D. and Tittle, E., 2002, **Scene of the Cybercrime**, Syngress, US

National Statistics, 2004, on-line at http://www.statistics.gov.uk/statbase/ssdataset.asp?vlnk=8427&More=Y, accessed 22nd Sept 2004

Netcraft, 2004, **On line server survey**, on-line at http://ww.netcraft.com, accessed 5ht August 2004

NIST, 2001, **General Test Methodology for Forensic Tools**, on-line at http://www.cftt.nist.gov/Test Methodology 7.doc, accessed 20th August 2004

NIST, August 2002, **Test Results for Imaging Tools**, on-line at http://www.ncjrs.org/pdffiles1/nij/196352.pdf), accessed 20th August 2004

NIST, 2004, **National Software Reference Library**, on-line at http://www.itl.nist.gov/div897/docs/nsrl.html, accessed 3rd July 2004

NIST, 2004, **Special Database 28**, on-line at http://www.nist.gov/srd/nistsd28.htm, accessed 4th July 2004

Norton, P. and Wilson, R., 1985, **The New Peter Norton Programmers Guide to the IBM PC & PS/2**, Microsoft Press, Washington, US

Middleton, B., 2002, **Cyber Crime Investigator's Field Guide**, Auerbach,

Oliver P., 2004, **Wotsit's Format, The Programmers Resource**, on-line at http://www.wotsits.org accessed 1st July 2004

Patzakis, J., 2002, **The EnCase Process, part of The Handbook of Computer Crime Investigation**, Eoghan Casey, 2002)

PCBiography,2004, **PCBiography**, on-line at http://www.fortunecity.com/marina/reach/435/,

Pfisterer, C., 2004, **Test Images and Procedures**, on-line at http://disktype.sourceforge.net/fss/ , accessed 26th August 2004, A program, written in C to determine the type of format used in a disk image.

Quotes for All, 2004, on-line at http://www.quotesforall.com/o/olsonken.htm, accessed 20th August 2004

Rouissev, V. and Richard, Dr G., 2004, **Breaking the Performance Wall: The Case for Distributed Digital Forensics**, On-line at http://www.dfrws.org/bios/day2/D2-RichardIII-Perf.ppt, accessed 14th September 2004

Sammes, T., and Jenkinson, B., 2000, **Forensic Computing – A Practitioner's Guide**, Springer, London

Samples, K., 2004, **Forensic Application of Infrared Spectroscopy**, on-line at http://www.arches.uga.edu/~kevins/founding%20fathers%20page.htm, accessed 24th Sept 2004

SearchStorage.com, 2004, on-line at http://whatis.techtarget.com/ateQuestionNResponse/0,289625,sid5_cid535522_tax286191,00.html, accessed 24th Sept 2004

Smith, M. F., 1991, **Software Prototyping**, McGraw-Hill, London

The SysOp, 2004, **The Obsolete Technology Website**, on-line at http://www.oldcomputers.net/, accessed 19th September 2004

US Government, 2004, **The Digital Millenium Copyright Act 1998**, on-line at http://www.copyright.gov/legislation/dmca.pdf, accessed 23rd September 2004

Vacca, J., 2002, **Computer Forensics Computer Crime Scene Investigation**, Charles River Media, Hingham, Massachusetts

Vaswani, V.,2002, **XML and PHP**, New Riders, Indianapolis, US

Virgina Tech, 2003, **Terascale System X**, http://www.tcf.vt.edu/systemX.html, accessed 26th September 2003

White, D., 2004, **Digital Forensics Using Hashsets National Software Reference Library**, on-line at http://www.nsrl.nist.gov/documents/hashapalooza2004/slides/index.html, accessed 23rd Sept 2004

Zakon, R., 2004, **Hobbes' Internet Time Line** on-line at http://www.zakon.org/robert/internet/timeline/, accessed 23rd September 2004

Appendix A

Correspondence with Dr Golden G. Richard III

From: "Golden G. Richard III, Ph.D." <golden@cs.uno.edu>
To: nick@microtechnical.co.uk
CC: "vassil >> \"Vassil Roussev\"" <vassil@cs.uno.edu>
Subject: Re: Distributed Digital Forensics

Great to hear from you. We too see a huge void in this area--I don't
know of any other work (aside from yours, now). It's possible that
something is happening internal to the intelligence community that we
don't know about, but as for academic research, I believe your group and
ours are all there is.

It's tough not having a single reference--people tend not to believe
that you've searched thoroughly. Glad to help. :)

I've cc:ed Vassil Roussev, who is also working on this project.

--Golden

Nick Pringle wrote:

> Hi
>
> I'm currently doing an MSc in Computer Crime at Glamorgan University in
> the UK. For my project I'm working on the idea of using XML to
> distribute forensic data over a wide area network. I started this in
> June and could not find anything, anywhere about this. I'm in my last
> week of the project before handing it in on Sept 30th and I thought I'd
> have one last search and found "Breaking the Performance Wall". As you
> can imagine part of my project submission is a literature search, as it
> stands I had nothing to report up to about 30 minutes ago and now I've
> got you as a reference. Hip-hip-hurrah!!
>
> Is this virgin territory as it appears? Or is there a dark clandestine
> group who know everything and just don't want me in.
>
> It's not finished yet but you can see where I'm at if you want to look
> at www.mscproject.org.uk The major fundamental difference it that I am
> assuming data transfer over slower Internet communications because I
> believe transporting images of 500GB drives on multiple DVDs and tapes,
> perhaps all over the world will become a problem as more and more
> interested parties want access to evidence. In 3.2 you do not foresee
> the communication leaving the private LAN, I do and hence my work is in
> XML.
>
> Regards
> Nick Pringle

--
Golden G. Richard III, Ph.D.   Dept. of Computer Science
Associate Professor, Jazz Consumer University of New Orleans
golden@*DIE-SPAMMER*.cs.uno.edu   Got Jazz?

Correspondence with Brian Carrier author of Autopsy

On Sep 19, 2004, at 10:50 AM, Nick Pringle wrote:

> Hi
>
> 1 I note you wrote your own web server, why? Security? I'm using
> Apache/PHP/MySQL do you consider this generally unsafe?

No, I was going for ease of installation. The web server part is a very minimal part of the tool and it was easier to include it instead of having people configure Apache to point to the right places, provide access control etc. It started off as a CGI script for Apache, but I found this was too complex and people were too afraid to be always running Apache when the weren't using the tool.
There is a lot in Apache that is not needed by Autopsy and it would be difficult to have people (such as law enforcement) to remove the unneeded components to reduce the security risk.

> 2 You have used pure html and actively warn people about having
> Jscript & VBScript active on their browsers. I'm aware of the security
> reasons for this. Do you think the limitations of html in the 'low'
> quality of the user interface is worth the extra portability &
> security?

Basic HTML has its place. I am not against making a more advanced interface that is less portable. I am just not a GUI person and Autopsy was a quick fix that has gotten much bigger then it was originally intended to.

> 3 Have you heard of anyone else doing work on my sort of project?

The CTOSE Project at the European Commission was working on a work flow-based tool that used XML to store all of the data. I never saw the final product and am not sure how related it is.

brian

From: Nick@microtechnical.co.uk
To: Brian Carrier 'carrier@cerias.purdue.edu'

I'm currently doing an MSc in Computer Crime at the University of Glamorgan in the UK. My MSc project is an investigation into the idea of a remote client/server style forensic tool based on XML. It's got a central evidence store where evidence is pre-processed against NIST database, the results stored in a MySQL database and is then available, at file and sector level, for access via XML. I've finished off this stage with a couple of VB clients but think Java would be far more suitable as it is non-platform dependant.
I have noticed Autopsy and have a copy working on my server. I wonder, would you mind answering a couple of questions (keep your answers brief, I don't want to waste your time too much).

1 I note you wrote your own web server, why? Security? I'm using Apache/PHP/MySQL do you consider this generally unsafe?
2 You have used pure html and actively warn people about having Jscript & VBScript active on their browsers. I'm aware of the security reasons for this. Do you think the limitations of html in the 'low' quality of the user interface is worth the extra portability & security?
3 Have you heard of anyone else doing work on my sort of project?

My ongoing work is available on http://www.mscproject.org.uk/ if you're interested.
Thanks very much for your time.

Regards
Nick Pringle