Information Exchange between Resilient and High-Threat Networks: Techniques for Threat Mitigation

Tim Dean and Graham Wyatt
QinetiQ – Trusted Information Management
Woodward Building B006,
St Andrews Road, Malvern,
WR14 3PS
United Kingdom

tbdean@qinetiq.com / gwyatt@qinetiq.com

SUMMARY

High resilience military networks frequently have requirements for exchange of information with networks of low assurance, including networks of unknown threat such as the Internet. Traditionally, the approach to solving this problem is an air-gap between the two domains, with information exchanged between them on floppy disk. This approach is both time-consuming and potentially risky. This paper proposes alternative techniques to enable assured, two-way, information flow between high resilience networks and other networks of unknown threat. The techniques include conventional and novel technologies designed to control and constrain information formats, manage the environment between domains with network level controls, and provide assured, user-instigated, release sanctions.

1.0 INTRODUCTION

Access to the Internet for e-mail and web access is now essential for military and government users. There are other, innumerable, unclassified or low classification systems with which military and government networks must also communicate, such as the Red Cross, the Red Crescent, police forces, other civil agencies, other government departments, and broad coalition networks.

There is inevitably a requirement to transfer information between such low classification systems to higher classification military systems that require higher assurance and a greater degree of resilience, such as command and control systems, logistics systems and intelligence networks. In such a context, Internet-connected systems are inevitably considered a high threat. Strict controls must be placed at the boundaries between these systems to prevent both the introduction of malicious content into the high system and the leakage of high data to the low system. *Note on terminology: from here on we refer to the two kinds of network as 'High' and 'Low Assurance'. Lower assurance arises due to the higher (or unknown) threat level, and in general is likely to lead to lower levels of resilience in applications.*

Historically, the security separation problem has been solved by total electronic separation between the low and high networks, sometimes referred to as an "air gap". But experience has shown that this is not always practical, as the low classification information may have high value in the high system – for example, weather data, situational awareness information, open-source intelligence, collaborative planning, and information from civil agencies.

There is a common requirement to enable a secure connection between these high and low networks in such a way that information can be exchanged without posing a threat to the resilience of the high network.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

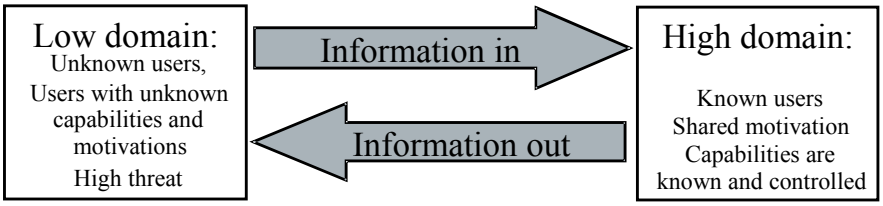

Figure 1 – Exchanging information between low and high domains

As a consequence, floppy disks, USB memory sticks and CD-RWs are often used to bridge the gap. But these techniques raise a number of risks and are inadequate to deal with modern threats that use multiple propagation mechanisms to gain access to networks. This paper will examine the threats of connecting low and high assurance domains, discuss some traditional methods of mitigating the risks posed by the threats and outline novel technical solutions that enable information exchange in particularly high-threat scenarios.

2.0 THE TECHNICAL THREAT

Some of the dangers posed to a high assurance domain are reported widely and frequently. For example: attacks against Internet facing web servers to extract client credit card details; e-mail arriving from the Internet carrying viruses that infect a business' Intranet; or more recently, the emergence of 'phishing' where the attacker masquerades as an on-line retailer or bank, sending e-mails to customers inviting them to click on a web hyper-link under the pretext of performing a necessary administrative task. The attacker, who owns the web site, can then harvest the usernames, passwords and credit card details of the victims and use these credentials to gain access to the high assurance system.

Attacks can be categorised in many ways, using terms such as:

- Viruses – malicious code that replicates itself to other host programs; areas of memory; disk boot sectors; or macro capable documents. Can also execute a malicious payload.
- Worms – malicious code that makes copies of itself and can exploit program vulnerabilities to propagate or can include the propagation mechanism within the code.
- Trojan Horses – a program that does not replicate itself but can damage the host computer or use the host to launch further attacks, often under the direct control of the attacker.
- Other kinds of attacks include: back doors, rootkits, BIOS and Microcode malware, social engineering attacks, and buffer overflows.

Anyone who reads the technical press will be familiar with examples of the above attacks and may well have been subjected to some of them. However the attacks that get reported are generally the opportunistic type that become global phenomena, such as variants of the Sobig or LoveBug viruses. What is less reported and less understood are the direct, targeted attacks. It is difficult to know if their apparent lack of frequency is due to their rarity, or the unwillingness of organisations to discuss such attacks publicly, or, more worryingly, their success (i.e. the attack succeeds and is so carefully disguised, it is never discovered).

The scattergun approach taken by opportunistic attackers is a time-consuming and troublesome nuisance to system administrators who must secure their Internet facing networks from the attacks. When these networks are connected to an affiliated high domain with high-resilience requirements, the high domain administrators must deal with the same threats posed by the opportunistic attackers, as well as potentially more devastating targeted attacks launched by skilled, motivated and highly funded attackers.

It is well known that the majority of attacks on the Internet are from relatively unskilled attackers making use of publicly available tools and code that exploit known vulnerabilities, for which manufacturers' patches are commonly available. A properly resourced attacker could discover new vulnerabilities, develop new exploits, and attack a network using these novel methods that would avoid detection by commercial intrusion detection and filtering systems. Such systems generally detect only known recognised patterns, or signatures, of attack.

A successful attack will compromise one or more elements of the information security trinity: i.e. information confidentiality, integrity and availability.

- Confidentiality – Information within the high domain should remain within the high domain unless its release to a lower domain is authorised and appropriate. For example, personal information relating to the clients of a bank must remain confidential and not be leaked to the Internet, either accidentally or deliberately.
- Integrity – Information within the high domain should remain uncorrupted. An integrity attack might lead, for example, to a message 'Credit Joe Bloggs £3000.00' being changed to 'Debit Joe Bloggs £3000.00.'
- Availability – Information services within the high domain must remain available. A well-known availability attack is the Distributed Denial of Service attack that has affected many Internet facing companies such as on-line banks and retailers. The servers of these companies are bombarded with bogus requests from thousands of computers infected with trojans controlled by the attacker. Valid user requests are unable to reach the server due to the overwhelming quantity of bogus traffic.

Analysing some recent, successful and well-publicised attacks reveals a new trend for *combination malware*, or to put it another way, malicious code that:

- Has combined characteristics of virus and worm for propagation, using mobile code (active content) that usually requires some manner of user interaction, as well as an element involving network attack that can happen automatically.
- Can embed itself into a system as a Trojan horse.
- Can add a back door allowing a two-way communication channel back to 'base' (or more likely, a web site or IRC chat room in the attacker's control) where commands or updates to the virus code can be posted.

An interesting point about such code is the diverse range of attack vectors being used for propagation. Methods include direct communication using TCP/IP, application channels such as SMTP, and corrupted application data, all being used in combination. Also of note is the fact that this kind of attack can propagate using floppy disks and CDs, therefore crossing what are perceived as "air gaps" to threaten the resilience of critical networks.

The recent Sobig.F (August 2003) attack exhibited many such characteristics. It used several propagation techniques, including spreading via e-mail attachments and network shares, and it included an embedded SMTP engine. The payload was an URL downloader which, in effect, meant that the payload was infinitely variable, being dependent upon the imagination and talent of the attacker to create their preferred attack and post the code to the web site from which the virus was pre-programmed to seek the download. The most obvious effect of the Sobig.F attack was denial of service through network flooding and e-mail system overload.

Furthermore, the number of malware attacks reported is increasing at an alarming rate. For example, CERT reports the following figures:

- 21,756 viruses reported in 2000
- 114,855 viruses reported in 2003 (to October)[1]

The propagation speed is also increasing as the number of attack vectors has increased, and as the time between the announcement of a vulnerability and the associated virus release has narrowed, sometimes to a few hours.

Yet, as previously discussed, these public statistics largely overlook the issue of targeted attacks, the prevalence and effects of which are almost completely unknown. Unfortunately, the threat of targeted attacks is the key concern to military networks.

3.0 SECURING THE BOUNDARY BETWEEN HIGH AND LOW ASSURANCE DOMAINS: COMBINING TRADITIONAL AND NOVEL COUNTER-MEASURES

As part of the research on behalf of the United Kingdom Ministry of Defence, QinetiQ has been examining how traditional and novel technologies can be employed to enable a controlled, bi-directional exchange of information between high and low domains.

We define three categories of techniques that can be used to control information flow:

- Format control
- Environment control
- User control and release sanctions

3.1 Format Control

Data format control techniques involve three related processes: (i) using 'safe' formats, (ii) format conversion - where data is transformed from one format to another format that obeys a different format 'grammar' and 'syntax' and (iii) checking that the formatting rules have been obeyed.

3.1.1 'Safe' Formats

The dangers of complex data formats are well understood and documented. For example, HTML used in web pages and e-mail can contain active content described with a scripting language that, by default, is processed and rendered by the client machine. Many attacks exploit vulnerabilities in the script interpreter or the host application to run malicious and damaging program code[3]. Similarly, complex formats used to describe documents created with a word processor can contain macros that again perform malicious actions. Word processor formats also allow for 'hidden content', where multiple versions of the document are embedded within the document description, containing data previously deleted by the user and not visible on the screen, but still encoded within the file[2].

Formats that ban or tightly constrain active content are safer and are in general to be preferred. One example is earlier versions of PDF (PostScript Distribution Format), which tightly constrained the use of scripting, banning dangerous functions such as general file access. However, these cannot be regarded as entirely safe since any complex language allows for the possibility of malformed data structures; this can produce unexpected behaviour in end systems, such as denial of service or buffer overflow attacks, which in effect convert a passive data structure into active code.

PDF suffers from such problems, and there have been some documented examples of such abuse[4]. Complex protocol specification languages such as ASN.1 (Abstract Syntax Notation One) also suffer similar problems. Projects such as Oulu University's Protos toolkit[5] have shown just how vulnerable ASN.1-based protocols can be to attack[6].

An approach to solving these problems is to write document viewers and protocol implementations that are carefully crafted and exhaustively tested. In general, industry is increasingly recognising the need for this, but complex languages cause the data-space requiring testing to be huge, or unbounded. Therefore, the assurance of such applications is limited, and something further is required in high assurance environments. A 'safe', or at least safer, format should be incapable of such (albeit unintended) subterfuge, using simple data structures with a well-defined syntax and grammar with a finite space.

Two obvious examples of simple formats are ASCII text and (certain kinds of) Bitmap images. Their simplicity aids another part of format control, the format checking. But before that happens, the data formats must be converted. This is discussed in the next section.

3.1.2 Format Conversion

The simple idea behind format conversion is that by transforming the representation of data from one form to another, possibly with several iterations, any malware capabilities contained in the original information format would be lost, especially if one of the transformation processes introduces an unpredictable, random element to the conversion process.

There are two stages at which format conversion might be used in a high assurance scenario explored in this paper.

First, many complex formats can be converted into ASCII text or Bitmap images. Every time we use the 'Print Screen' function in Windows (Press Alt key and Print Screen key together) we convert whatever is on the screen into a Bitmap image (which is then copied to the Windows clipboard). To take one example, if we take a Bitmap image of a Word document, all the informational content remains (we can still read the document), however all the hidden content encoded in the complex Word format is removed, along with the associated threat posed by that hidden content.

There are a number of commercial programs that can be used to take Bitmap images of complex documents (in HTML, Word, Excel, Adobe Acrobat, and other formats) in a more sophisticated manner than a simple 'Print Screen' request. For example, these 'screen-scrape' programs can open a Word document so that the entire document is converted to a Bitmap image, not just the single page visible on the screen.

In addition, some of the screen-scrape programs incorporate a 'text-scrape' function where the text within a Word or Adobe formatted document can be converted to ASCII text, again removing any hidden content (though at this stage any text-scraped scripting code, for example, may be preserved).

It is worth differentiating between the effect of a screen and text scrape. The text scrape takes a document and translates the complex encoding used by the word processor to represent letters of the alphabet and translates that into simple ASCII encoding. However both encodings are used to represent letters of the alphabet. Converting a word-processed document into an image profoundly alters the encoding; while the word processor is encoding letters of the alphabet, the image is encoding the colour and brightness of pixels in the image. It is only in the viewer's brain that those encodings are translated back into text.

The second time a format conversion might be performed is when releasing Bitmap images, just after the format checking is completed. At this stage, the Bitmap image could be transformed into a JPEG image.

The transformation process, using a lossy compression algorithm like JPEG, would mean that while the information content would remain the same (the JPEG and Bitmap images would appear almost identical to the viewer), the data format would undergo a transformation.

This second transformation would make it even less likely that an opportunistic attacker's hidden or malicious content would be preserved, particularly given it may be irretrievably lost thanks to the lossy compression. Compression is of course doubly desirable given that bandwidth restrictions would make the exchange of large Bitmap files difficult to scale to a large user community.

To defeat the targeted attacker it would also be possible to introduce some random alterations to the Bitmap image prior to the (optional) JPEG conversion; for example, by making small, random modifications to the brightness values of some, or all, of the pixels in the Bitmap, and even to the image dimensions by adding a randomly sized border to the image. The changes would be near undetectable to the viewer of the final image, but make it very difficult for an expert attacker to predict the exact output of the conversion and so design an attack to exploit a hypothetical vulnerability in the recipient's image viewer.

Crucially, the initial screen or text scrape conversion process does *not* need to be trusted or assured. What *must* be assured is the optional process to randomise Bitmaps and the format checking stage, described below.

3.1.3 Format Checking

There are an increasing number of products that are designed to check complex formats such as HTML, XML, Word, or Excel for hidden content, malicious macros and the like. Such checkers are extremely useful and valuable in scenarios where this richness of information transfer is permissible. They can use simple heuristics such as ensuring that Word documents have not been "fast saved". But simple heuristics are not adequate to address the kind of threats discussed above.

For these situations, much more constrained data formats are required. If only two simple information formats such as ASCII text and Bitmap images are allowed to be exchanged between the two domains, the scale of the problem is much reduced.

By only permitting ASCII text messages and the simplest variety of bitmap images to pass between domains there is a significant reduction in the complexity of the content checkers. For example, an ASCII text checker would consist of one very simple function that would eliminate from the message any characters that were not from a recognised alpha-numeric list or from a small subset of permitted punctuation markings. The punctuation could even be replaced by words, as in old-fashioned telegrams (STOP for fullstop). All other content would raise a warning and be removed from the message before being passed on. Therefore if the message contained Javascript, the opening and closing brackets ('<' '>') used to denote a 'tag' would be removed, rendering the Javascript incomprehensible to the receiving application. An optional, simple ASCII text search could warn of the presence of blacklisted words.

A Bitmap checker would be simpler still, as long as the simplest kind of raw Bitmap format is used exclusively. These simple bitmaps are a rudimentary image format to parse, containing a definition of the dimension of the image (numbers of pixels across and high), followed by the colour parameters for each of the pixels (defined in terms of the strength, from 0 to 255, of the Red, Green and Blue constituent colours). The Bitmap format checker would perform a simple grammar and syntax check of each image so that each file was known to conform to these Bitmap encoding rules.

3.2 Environment Control

The techniques of format control discussed above must be carefully managed and protected to ensure that they cannot be subverted. Here we specifically refer to the environment at the interface between the high and low assurance domains. Traditional techniques to control the interface environment include firewalls, one-way data diodes (where information can only flow in one direction, for example from a low to high domain), and the use of De-Militarised Zones, generally bounded by two firewalls or data diodes (or both).

To construct such a controlled environment, QinetiQ has used commercial off-the-shelf components, some of which have formal (EAL) assurance rating. Some of the components are QinetiQ products, such as the one-way data diode (SyBard/Diode)[1] and the SWIPSY® firewall toolkit. The SWIPSY toolkit is an E3 (equivalent to EAL4) evaluated product that allows additional code to be added to its security 'compartments' without affecting the evaluation status of the toolkit itself. SWIPSY has security properties that assure network and process separation: processes communicating with one network can not communicate directly with the other network other than by 'trusted mover agents' that in turn force data to be passed to the format and content checkers.

Environment controls are insufficient individually to control the new generation of malware, as the controls typically defend at the network level. However, when combined with application level 'checker' software and the novel techniques discussed in this paper, they are vital in forcing the data through the checking processes. A number of other mechanisms must also be used if high assurance is required. These include assured user sanction, and potentially, machine virtualisation and these are discussed in the following subsections.

3.3 User Sanction and Control

The assured intervention of a human user is critical for the release of data from a high to low domain for two reasons. First, if every exchange of information from high to low is governed by user sanction, the process can be meaningfully audited, making the users accountable for their actions. Secondly, a properly implemented and assured user sanction mechanism would prevent any high machine infected by a back-door attack from communicating with the low domain by any other route than the user sanctioned channel. With this restriction on information flow in place, as well as the format checking and format conversion, it becomes a significantly challenging task for the back door program to communicate with the low domain attacker without arousing the suspicions of the sanctioning user.

The software used to implement the user sanction process should be simple and trusted. First, the data should be presented to the user for approval and release. This data should be confined to simple formats such as ASCII text or Bitmap images as previously discussed. Second, the user could, optionally, instruct the software to digitally sign the data to prevent modifications between the user's desktop and the domain boundary. The user must be able to view everything on the screen that they are about to sign. This implies that a trusted viewer is used, i.e. an assured computer that the user can trust will display the complete contents of the message and that this viewer and signer can not be subverted by any attack.

It is worth mentioning that the idea of a trusted signing device is a recognised requirement in some civil applications. The EU Digital Signature directive[7] recognises the need for a Secure Signature Creation Device (SSCD), and some EU member states have enshrined this requirement in their legislation. In addition, industry seems to be following a similar trend: the Trusted Computing Group's (TCG)[8] Trusted Platform Module (TPM)[9] specification includes "Data Attestation" facilities for signing data structures. The most recently published documents describing Microsoft's Next Generation Computing Base (formerly Palladium) detail aspirations for both trusted viewers and trusted signing mechanisms, as part of the trusted part of the operating system[10].

[1] **Part of the QinetiQ SyBard::Suite®. SyBard::Suite and SWIPSY are registered trademarks of QinetiQ Ltd.**

4.0 HIGH ASSURANCE PROTOTYPE – SENDING INFORMATION FROM HIGH TO LOW DOMAINS

As part of its research, QinetiQ has experimented extensively with prototype implementations of the above techniques. The prototype mechanism for releasing information from high to low domains is described in the figure and text below.

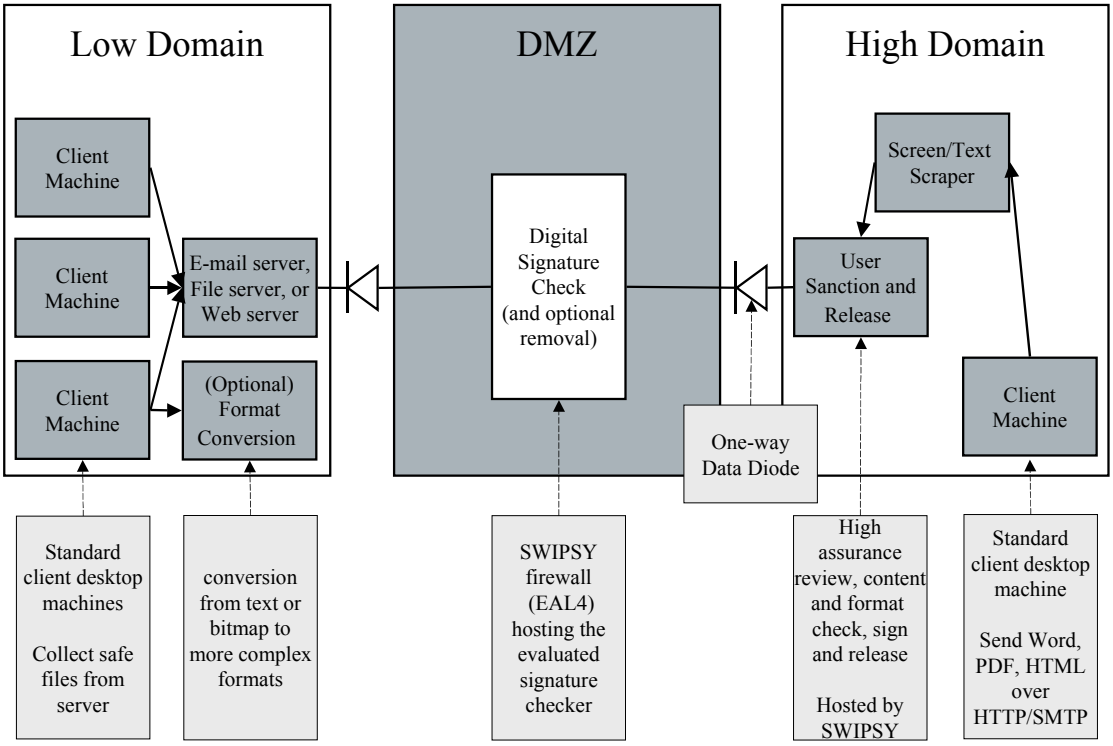

Figure 2 – The High Assurance prototype mechanism for High to Low information release

The steps in the release process are as follows:

- Format Conversion: The user connects to a high domain web server that can perform the screen-scrape or text-scrape, converting the complex file (e.g. a Word document) into a Bitmap image or ASCII text document
- The converted file is routed, via e-mail, to the user sanction release mechanism
- The trusted user sanction device is built upon the Trusted Solaris operating system (EAL4 rating) modified with the SWIPSY toolkit (EAL4 rating also). Upon this system a simple console-based viewing program has been written and installed
- The console viewer strips the MIME encoding surrounding the message body (or attachment in the case of a Bitmap image), performs the format check, and presents the ASCII text or Bitmap to the user
- The user can then choose to stop or release the message. If the release option is chosen, the message is digitally signed and packaged in an S/MIME envelope.
- The digital signature is applied using an assured cryptographic library (currently the CESG Cryptserve algorithm suite - EAL4) and, in this early prototype, a full S/MIME toolkit. Future versions of the system will employ ‘cut-down’ program code to create the S/MIME envelope. One

slight modification has already been made to the S/MIME standard: The inclusion of a copy of the "To:", "From:" and other header fields in the message body in order to counteract the subtle S/MIME weakness where header information is unsigned. However, this does not affect its correct reception and checking in a standard S/MIME client.

- The signed message is sent to the DMZ and to another SWIPSY firewall (this one operating without human intervention). At the DMZ firewall the digital signature is verified and (optionally) removed before it is forwarded to the low domain SMTP server.

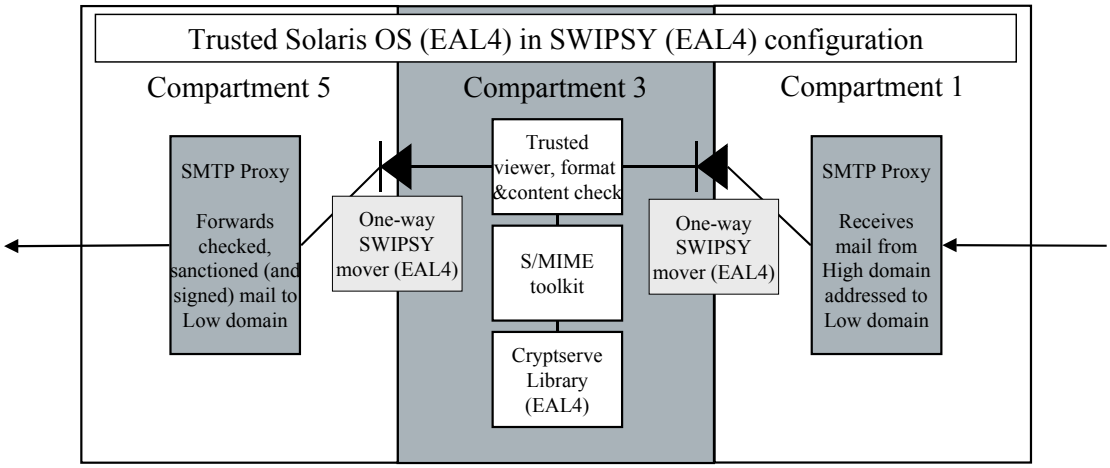

Figure 3 – Detail of the User Sanction and Release Mechanism

5.0 HIGH ASSURANCE PROTOTYPE – SENDING INFORMATION FROM LOW TO HIGH DOMAINS

The route from a low to high domain does not necessarily require a user sanction (although there may be good security reasons why such intervention may be required in some circumstances). This simpler process can be fully automated and is detailed in the proposed chain of events and diagram below:

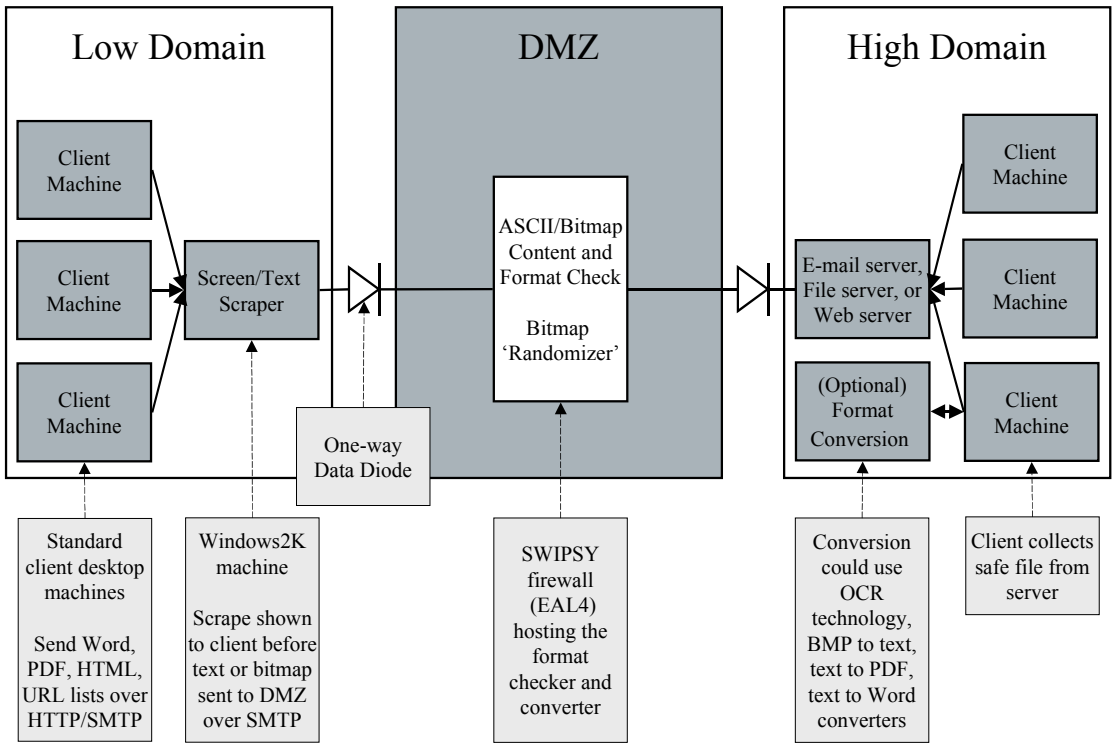

Figure 4 – Sending information from low to high domains

- Format Conversion: The user connects to a low domain web server that can perform the screen-scrape or text-scrape, converting the complex file (e.g. a Word document) into a Bitmap image or ASCII text document
- The scraped file is sent through the one-way data diode to the trusted format checker hosted by a SWIPSY machine in the DMZ. The checker performs a content check on the ASCII or Bitmap file as before, and if successful…
- The Bitmap file is 'randomised' and optionally transformed into a JPEG file
- The transport protocol encodings that surrounded the file are stripped and reformed on the SWIPSY machine (e.g. MIME encoding and SMTP headers). Alternatively, protocols like FTP can be used that require no transport 'envelope' around the file.
- The checked file is passed to the high domain through a second one-way data diode
- The file is delivered to the high domain server from which the client can collect the 'safe' file
- The file *could* be transformed into the original document format at this stage, for example into a Word file. This is a potential problem though as the original 'malware' may be reconstructed as part of the process

It can be seen that the data diodes illustrated in the three diagrams above are pointed in different directions, creating a two-way flow of data between high and low that might appear to make the data diodes redundant. However, the high and low domains illustrated in the diagrams are generic labels and not meant to imply that the low domain is the same low domain in both diagrams. Rather, the scenarios described in this paper are assuming the high domain has a multiplicity of connections to lower domains.

6.0 FUTURE DEVELOPMENTS AND ALTERNATIVE DESIGNS

6.1 Transforming Complex Files to XML

This paper has explored a scenario requiring particularly stringent constraints on the information transfer between domains. Other systems may require a 'richer' exchange of information, involving more complex file types and data encodings than simple ASCII text and Bitmap images. Examples include PDF files and database updates. For these kinds of files, QinetiQ is developing techniques for transforming complex file types into a single, standard XML representation. The complex (but uniform) XML would then be subjected to content checking on a trusted platform, and then transformed from the XML back to the original encoding (Word again) or to a new encoding, such as PDF format.

6.2 Trusted Viewing and Signing on the Client Desktop with Virtual Machines

Machine Virtualisation allows a second complete operating system to be installed on a user's machine and for that second operating system to run concurrently as a second 'virtual' machine with the host operating system. Virtual machine software such as VMWare allows many such virtual machines to operate simultaneously.

There have been some interesting developments recently using virtual machine technology to create secure multi-system desktops where the two virtual machines are separated by assured mechanisms (for example, the NetTop approach developed by the NSA[11]).

It would be advantageous if the trusted user sanction mechanism could be built into a separate virtual machine on the client's desktop. This could then involve an assured and highly locked down Operating System that is specifically for this one purpose. Such a development would add flexibility and convenience, allowing users to release documents from their own desktops, either to replace the trusted 'domain' signer described in this paper, or in addition to their sanction, creating an effective two-man rule release mechanism.

7.0 CONCLUSIONS

This paper has proposed a combination of techniques to allow a potentially assured, two-way exchange of information between high and low assurance domains. The scenario has assumed that the stringent requirements for separation of the two domains could only be achieved by an air-gap, given current commercially available solutions.

What has been proposed in the paper is intended to offer *greater* security than an air-gap, principally because, like winning the lottery, an air-gap is an attractive idea in theory but difficult to achieve in practice. Pragmatic users faced with an air-gap between security domains will often reach for a floppy disk or USB memory stick rather than re-key the entire document from the high machine to the low.

The combination of format controls, environment controls, and user controlled release sanctions outlined in this paper have been designed to offer a secure solution to the urgent and unavoidable business requirement to share information.

8.0 REFERENCES

[1] CERT Co-ordination Statistics 1988-2003, http://www.cert.org/stats/cert_stats.html

[2] Simon Wiseman, QinetiQ White Paper, “Documents with Hidden Surprises”, http://www.qinetiq.com/homepage/services/information/white_paper0/documents.html

[3] George Guninski, Security Research Web Page "Internet Explorer Security", http://www.guninski.com/browsers.html

[4] Common Vulnerabilities and Exposures, “Buffer Overflow in Adobe Acrobat…”, http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0713

[5] Univeristy of Oulu, Security Testing of Protocol Implementations, http://www.ee.oulu.fi/research/ouspg/protos/

[6] Kevin Poulson, The Register, "Brits pound OpenSSL bugs" http://www.theregister.co.uk/content/archive/33148.html

[7] Directive 1999/93/EC of the european Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures. Official Journal L 013, 19/01/2000 p. 0012 – 0020. http://europa.eu.int/ISPO/ecommerce/legal/digital.html

[8] Trusted Computing Group Homepage, https://www.trustedcomputinggroup.org/home

[9] Trusted Computing Group, Trusted Platform Module, "Design Principles", https://www.trustedcomputinggroup.org/downloads/tpmwg-mainrev62_Part1_Design_Principles.pdf

[10] Microsoft, Next-Generation Secure Computing Base Homepage, http://www.microsoft.com/resources/ngscb/default.mspx

[11] National Security Agency, NetTop Technology Profile Fact Sheet, http://www.nsa.gov/programs/tech/factshts/20030103-3.htm