Applying Military Grade Security to the Internet

C. I. Dalton <cid@hplb.hpl.hp.com>
J. F. Griffin <jfg@hplb.hpl.hp.com>

Abstract

The explosive growth witnessed in the Internet over the last few years has encouraged companies to connect to it and to offer services to their customers over it. Concerns about security are holding them back from all but the most restrictive connectivity.

This paper explores the use of a military development, the Compartmented Mode Workstation, in a commercial setting, as a platform that is secure enough to implement services that are accessed over the Internet. Two applications have been investigated in detail, a firewalled Domain Name System and a World Wide Web service with enhanced authentication. Finally, there is discussion of how other Internet-based services might benefit from the application of CMW technology.

This work was carried out as part of the E2S project in the European IVth Framework Programme, IT RTD Project No. 20.563.

I. Introduction

The business community has not been slow in recognising the potentially vast commercial opportunities created by the remarkable growth in the Internet. Further, many businesses view Internet based e-commerce as critical to their long term survival. The fundamentally insecure nature of the Internet is, however, proving to be a limiting factor in allowing the maximum exploitation of these new found electronic markets.

Over the last three decades, military agencies have spent considerable amounts of funding on research and development of computer security, ever anxious to find ways of guaranteeing secure electronic communication across their own private networks. With the current desperate commercial need for network security, it would seem obvious to look at ways of applying the military generated technology in the Internet arena.

In this paper we present the results of our investigation into the use of one particular military development, Compartmented Mode Workstation (CMW)[1], in a role as an application gateway situated between internal systems and the Internet. We describe scenarios where use of CMW technology in this role can be advantagous, creating a firewalled Domain Name System service and providing enhanced authentication of World Wide Web services. We end with a discussion of how other Internet based services might benefit from the application of this technology.

II. The Compartmented Mode Workstation

The Compartmented Mode Workstation was originally developed for military and government use according to the CMWEC criteria[2] for evaluating trusted systems. The CMW class is an entirely separate but related set of criteria to the more familiar Orange Book criteria[3]. In Orange Book terms, CMW has all of the B1 level security features, and includes a number of B2 and B3 features. A number of the CMW features are relevant to Internet firewalls/application gateways, in particular, mandatory access control (MAC); discretionary access control (DAC); privileges; command authorizations; audit.

The combination of these security features makes CMW especially suitable as an application gateway. Some features make it easier to administer and maintain the gateway machine in a secure state and to detect attempts at attack: the detailed auditing, the command authorizations allowing separation of duty and retirement of the root account, and the trusted execution path combating Trojan horses. Other features make it possible to build and run applications securely: MAC and privileges in particular. This section explains these security features; the remainder of this paper concentrates on the use of these features to develop applications to run securely on CMW while providing access from the Internet to sensitive resources and information.

II.A Mandatory Access Control

Mandatory access controls are enforced consistently by the operating system - users cannot choose which information will be regulated. On CMW all information has associated with it a sensitivity label. The sensitivity label comprises a 'classification' and a number of 'compartments'. The operating system labels files, processes and network connections. In general, to have read access to some data, a process must have a sensitivity label which 'dominates' the label of the data. A sensitivity label is said to dominate another when its classification is higher or equal to the other's classification, and when it includes all compartments included in the other label. For write access, a process's label must exactly equal the data's label.

In practice, classification is generally used to indicate how secret or sensitive data is. Compartments, however, are often used to partition data so that access to separate sets of data is given to different groups of users, e.g., members of different departments in a company. The configuration shown in figure 1 below uses compartments to distinguish between data and resources accessible from the Internet, and those accessible from a company's internal LAN.

CMW supports MAXSIX trusted networking[4]. When communicating with hosts that are not trusted or do not support labelling, the system automatically attaches sensitivity labels to all packets arriving from or sent to the remote host. The label can be applied according to which interface card the packets arrived on or the Internet Protocol address of the remote host. This combines with the MAC features, so the operating system prevents the remote host communicating with processes at other sensitivity levels and accessing inappropriate information.

II.B Discretionary Access Control

CMW has discretionary access controls on similar lines to most Unix systems, including read-write-execute protection based on user and group id's and access control lists. The work described here concentrates on the use of mandatory controls; discretionary controls are less relevant. In particular, the DAC features seem less suitable precisely because they are under the control of the user.

II.C Privileges

On CMW, the root account's special powers are replaced by a large set of individual privileges. The relevant privilege is checked by the kernel whenever a process tries to make a system call which could in some way compromise security. There is a total of approximately 50 different privileges, ranging from fairly harmless to very dangerous.

Some of the most dangerous privileges are those which allow a process to override the MAC, and these must be carefully granted to allow selected traffic to cross the firewall. For safety, we grant privileges only to small relay programs which are specially designed and carefully reviewed. These 'trusted' programs allow information to cross compartment boundaries, so that large pre-existing applications can be safely accessed from sensitivity levels other than their own. The trusted programs must follow the 'least privilege' principle: they raise a privilege only while it is needed for a particular operation and lower it again immediately afterwards.

II.D Command Authorizations

Command authorizations are the sisters to privileges. They are given to users, whereas privileges are granted to programs. Authorizations allow control over which users are allowed to invoke which trusted programs. By allocating different sets of authorizations to different users, we can achieve separation of duties. No single user has absolute control of the system; rather there are a number of administrative roles with complementary powers.

II.E Audit

The trusted kernel audits system calls, and trusted applications can audit their own actions using a standard auditing subsystem interface. This auditing cannot be overridden without special privilege. It will normally be configured to log any access denial or insufficient privilege for an attempted operation. Trusted programs can log their actions directly in an easily understood form, so an administrator can track any suspicious behaviour involving overriding MAC without having to decipher long sequences of system calls.

II.F An Example Configuration

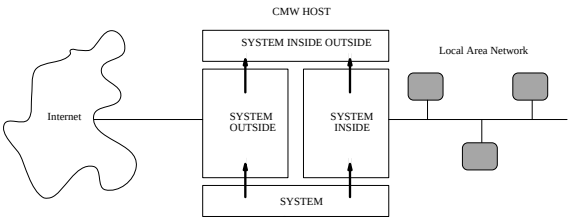

Figure 1: Simple CMW configuration

Figure 1 shows the permitted flows of information in a simple configuration. In this

example, there is a single classification level, SYSTEM, and two compartments, INSIDE and OUTSIDE. On such a system, a process labeled SYSTEM OUTSIDE would have read-write access to information labeled SYSTEM OUTSIDE, read-only access to information labeled SYSTEM and would have no access to any other information on the system. A process at SYSTEM INSIDE would have read-write access to an entirely separate set of information labeled SYSTEM INSIDE and read-only access to the information labeled SYSTEM.

As shown in the figure, this configuration can be used for a firewall host with two network interface cards, one connected to the external Internet and the other connected to an internal local area network (LAN). The MAXSIX networking is configured such that packets from the Internet are labeled SYSTEM OUTSIDE and packets from the internal LAN are labeled SYSTEM INSIDE. In this configuration, a connection from the Internet can communicate only with processes labeled SYSTEM OUTSIDE. Even if an attacker from the Internet were to gain access to a shell on the CMW machine, he could not modify files labeled SYSTEM, and could not access files or services labeled SYSTEM INSIDE.

III. The Domain Name System

One of the most fundamental components of the Internet infrastructure is the Domain Name System (DNS)[5,6]. DNS provides a mapping between host names and numerical Internet Protocol addresses. Although essentially only a user level service, without it much of what is carried out on the Internet would be impossible.

Whilst DNS is a vital service, the security problems posed by using DNS are numerous[7,8]. The main risk surrounds the amount of local information that the DNS can make public. Allowing access to a site's hostnames opens up the potential for attacks where name-based authentication, such as .rhosts files, is in use. It also provides a lucrative source of inside information for the social engineering type of attack. Finally, if local machines and domains are named after company projects, which is common, then the availability of this information to competitors via the DNS is something that an organization may want to block[9].

In light of these risks, many sites choose to operate a firewalled split DNS environment. A server in front of the firewall provides a minimal subset of the DNS records for the zone. Generally, only records essential to achieving successful communication with external systems are published. Example records would be MX records for mail routing and address records for local services that can be accessed externally, such as World Wide Web servers. Queries from the Internet only have access to this small set of data. Behind the firewall, another DNS server is run. This server contains all the data for the zone, thus allowing internal clients access to all the local host information.

Whilst a firewalled solution such as this succeeds in providing the required level of information hiding, it poses a further problem of how to allow internal clients the ability to resolve external addresses. A typical solution to this revolves around the use of two hosts and an internal packet filter between them[10]. By using CMW technology, however, the same protection from DNS attacks can be achieved with only a single host and without the need for complicated internal packet filtering. The use of a CMW host as a platform for a DNS service also protects against a name service configuration attack[11].

III.A A CMW-based DNS firewall solution

The following discussion surrounds the simple network architecture and CMW configuration of figure 1[1]. The aim of the CMW-based DNS service is to provide different DNS information to querying clients depending upon whether the query originated from the external network or the internal network. External clients should have access only to DNS records stored at SYSTEM OUTSIDE. Internal clients should have access to records stored at SYSTEM INSIDE. Additionally, the internal querying clients should be allowed to access to DNS information from name servers on the external network. External clients should not be able to access internal DNS information.

To provide this service a separate non-privileged name service daemon is run for each sensitivity level packets may be received at. In our simple example there are two name service daemons running, one at level SYSTEM OUTSIDE and the other at level SYSTEM INSIDE (figure 2). The two name service daemons are each authoritative for the same zone. However, since they are separate processes, they can contain different zone data. The name service daemon running at the level SYSTEM OUTSIDE has the minimal subset of DNS records, consisting of only those necessary for external systems to locate locally provided public services such as World Wide Web servers and mail gateways. Conversely, the name service daemon

[1] The ideas presented have been extended to more complicated configurations.

running at the SYSTEM INSIDE level holds a full set of DNS data for that zone. Internal querying clients therefore have access to all the DNS records for their zone but external clients have a much more limited view.

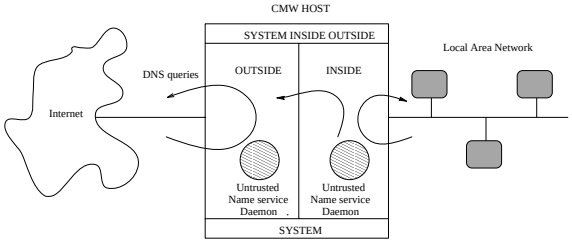

Figure 2: CMW multilevel DNS configuration

On CMW, network daemons without privilege can only listen for connections at one particular sensitivity level. There is also a restriction that only one process can listen on a particular port even if several processes at different sensitivity levels wish to. The approach taken here to provide a DNS service on the CMW to both the external and internal networks on the same standard DNS port(53) involves a small trusted front end wrapper daemon. This daemon has the privilege to allow it to listen for network connections at more than one sensitivity level, specifically SYSTEM OUTSIDE and SYSTEM INSIDE. The alternative would require DNS client software on either the external or internal networks to contact the DNS service at a port other than the standard one or to make the whole of the name server code trusted. Whilst adoption of the principle of least privilege under CMW helps, it was thought wise to still keep trusted programs as small as possible to help guard against exploitation of program bugs by an attacker.

When a DNS packet is received, it is simply forwarded by the front end daemon to the particular non-privileged name service daemon running at the sensitivity level of the incoming packet. The trusted front end daemon has the privilege to change its sensitivity level so that it can talk to processes running at a different level to itself. The non-privileged name service daemon is responsible for doing the actual resolving of the query. It is authoritative for the DNS zone formed by the local area network. Any communication it wishes to carry out on the network, such as contacting other name servers, must be proxied through the trusted front end wrapper daemon.

The front end wrapper daemon has also been trusted with the ability to selectively change its sensitivity level when forwarding queries on behalf of the untrusted name server daemons. This effectively allows the non-priviliged name servers to query other name servers on different networks to their own but always under the arbitration of the front end daemon.

In our simple example, the front end daemon would be configured to proxy packets for the SYSTEM OUTSIDE name service daemon over the SYSTEM OUTSIDE network only and to proxy packets at the request of the SYSTEM INSIDE name service daemon over either the SYSTEM INSIDE or SYSTEM OUTSIDE networks. This allows the internal name server to query other non-local name servers out on the Internet in order to resolve an external address for an internal client but blocks external clients from accessing the internal name servers.

III.B Implementation Details

III.B.1 The trusted front end wrapper daemon

The main body of the front end daemon (MLNAMED) is implemented as a continuous loop. It sits and waits for incoming DNS queries (both TCP and UDP) and for answers sent back from the untrusted name service daemons that require forwarding over the network to the original querying clients.

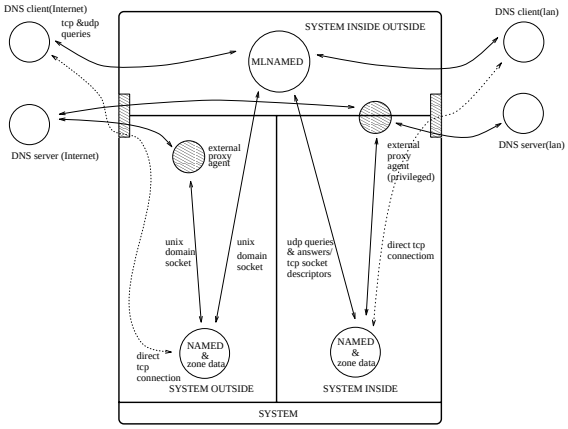

Figure 3: Detailed MLNAMED / NAMED architecture

For UDP queries, MLNAMED adds the hostname and port number of the querying client to the DNS query packet and passes it via a Unix domain socket to the name service daemon running at the incoming query level. Answers to the query found by the name service daemon are sent back down the same Unix domain socket to MLNAMED which then sends them on to the originating client.

In the case of TCP queries, after accepting the connection, MLNAMED passes the socket descriptor to the name service daemon at the incoming query level, again over a Unix domain socket. The name service daemon, because communication is required at only at one level, talks directly with the query TCP

client from then on. MLNAMED takes no further part in the process.

Figure 3 illustrates the structure of the CMW-based DNS in detail. The selective cross-compartment forwarding by MLNAMED is implemented via so called external proxy agents. At start up, MLNAMED spawns an external proxy agent for each of the network levels on the machine. The external proxy agent for a particular sensitivity level is marked as privileged or non-priviliged depending upon the configuration options provided to MLNAMED. These children handle the communication between the untrusted name service daemons and other, non-local, name service daemons. When an untrusted name service daemon needs to query a non-local name server in order to resolve a client query, it sends a query packet via another Unix domain socket to the external proxy agent at its own level. The external proxy agent checks which network level the packet would need to go out at to reach the non-local name server. If the level is the same as its own then it simply sends the packet out over the network, and passes any replies it receives back to the untrusted name service daemon. If, however, the level is different and the external proxy agent was marked privileged when it was spawned, the external proxy agent changes to the required level and sends out the query packet. Otherwise the name server query fails.

III.B.2 The non-privileged name service daemons

The non-privileged name service daemons are slightly modified versions of the standard Unix based BIND[2] NAMED servers[7]. They have been changed to listen and communicate over Unix domain sockets only, instead of directly on a network port. One Unix domain socket is used for receiving queries proxied via from MLNAMED and sending answers back, another socket is used for communication with other non-local name servers via the external proxy agent.

III.C Future

The deployment of IPv6 over the coming few years, and the application of digital signing to DNS records, will help make the use of DNS less of a security threat. However, the need for information hiding in the use of DNS is likely to remain.

[2] BIND4.9.3-beta24

IV. Secure User Sessions for WWW Applications

IV.A Background

The Authorization Facility described here was designed originally for a demonstrator being built by Hewlett-Packard's Customer Support Organization. Their system was to provide confidential support information to a group of partner firms over the Web. They wanted to authenticate their users and to initiate a secure session which could be tracked by their application. A user who was idle for more than 10 minutes would have to enter name and password again before continuing.

The application was implemented on the HP Virtual Vault platform[12]. Virtual Vault runs on CMW configured with two compartments, OUTSIDE and INSIDE, as described in the previous section. The web server runs at level SYSTEM OUTSIDE, so it can accept connections from the Internet. HTML documents are labelled at SYSTEM, so they can be read but not modified by the web server. Sensitive applications run at level SYSTEM INSIDE, so they and the data they manipulate cannot be accessed directly by the web server — or by an attacker connecting over the Internet. The applications are invoked through the CGI interface, by means of the Trusted Gateway Agent (TGA), a privileged program which invokes the applications on behalf of the unprivileged web server.

The Authorization Facility uses HTTP cookies[13] to provide secure sessions. The user is authenticated at the start of a session and sent an unforgeable ticket, the session ID, in a cookie which is attached to all subsequent requests from the user. The user is forced to reauthenticate himself at regular intervals or after a certain amount of idle time, at the choice of the application server. SSL[14] is used to provide condifentiality and integrity over the communications link. This has been designed as a general-purpose facility which can be applied to any web application.

IV.B Design for CMW

On the Virtual Vault, the HTTP basic authentication feature is turned off. It would require password information to be accessible to the web server, running at SYSTEM OUTSIDE. If an attacker were to break in, he might be able to read this and analyse it at his leisure. So all password information must be stored at SYSTEM INSIDE, where he couldn't access it. Neither can the web server, so there must be a trusted daemon which has enough privileges to do so.

Similarly, we do not trust software running at SYSTEM OUTSIDE to make or enforce authorization decisions regarding CGI applications running at SYSTEM INSIDE. Authorization must be carried out by software running at SYSTEM INSIDE, so it cannot be side-stepped by an attacker. Originally the Authorization Facility was implemented using functions integrated with the web server, and these are still used to control access to HTML documents stored at SYSTEM. Access to CGI programs is controlled by software invoked by the TGA, and running at SYSTEM INSIDE.

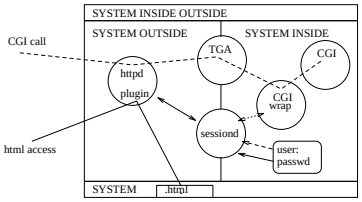

Figure 4: Configuration of Auth. Facility

Figure 4 shows the components involved in servicing an authenticated request and their sensitivity levels. Requests which invoke a CGI program are dealt with as normal by HTTPD; it invokes the TGA to have them executed. The TGA is configured to call a CGI "wrapper" rather than the original application CGI program. The wrapper checks for a session cookie and asks SESSIOND to validate the session ID. If the session is valid the wrapper executes the original CGI program; if not, it directs the client browser to authenticate the user.

Other URLs are checked using functions integrated with HTTPD. On each non-CGI request, HTTPD looks for a session cookie, makes a call to SESSIOND and either continues with the request or forces the user to be authenticated.

SESSIOND checks usernames and passwords, generates session identifiers, answers queries on sessions and applies the reauthentication policy. It is the only piece of trusted software that had to be added to the system. It has been given privileges to change its sensitivity level so that it can accept requests from the web server running at SYSTEM OUTSIDE and read the password file stored at SYSTEM INSIDE. It also has a privilege which allows it to turn off logging of system calls. Instead it can put its own entries in the system log files, which will be more comprehensible to the security administrator. SESSIOND needs to be privileged only if it is being called directly by the web server. If it is only being used to protect CGI programs, then it can run entirely at SYSTEM INSIDE, and there is no need to grant it any privileges at all, so there is one less piece of trusted software on the system.

SESSIOND can be run at SYSTEM INSIDE or it can have a compartment of its own. Running at SYSTEM INSIDE is simple and involves no changes to the sensitivity labels configured on the system. Giving SESSIOND a compartment of its own is potentially more secure and allows it to be accessed from multiple compartments in a more general CMW configuration. It could also run on an entirely separate machine, which would be normal for an authentication server in a non-CMW environment, but in this case it would require the addition of secure communication between the application server and the authentication server.

IV.C Attack Scenarios

Here are some types of attack and what protection the Authorization Facility running on CMW gives against them. In each case it is assumed that the attacker is connecting from the Internet, and so is at sensitivity level SYSTEM OUTSIDE. Should any one of these attacks occur, here is what would happen.

1. Attacker breaks into server and has access to a shell.

Without special privileges to override the mandatory access controls, the attacker can read HTML documents and general system files, but cannot modify them. Similarly, the MAC ensures he has no access to CGI programs or the sensitive data they manipulate.

2. Attacker finds a bug in HTTPD and can make it execute arbitrary code.

The attacker can access any HTML document and run any functions integrated in the web server, but without a valid session ID, any attempt to execute a CGI program will be rejected.

3. Attacker breaks HTTPD, intercepts a valid request and modifies it for his own purposes.

This is a more difficult attack than the previous one. It involves installing code in the web server process which does not stop it functioning, listens to requests going by, and modifies them. Such an attack could succeed.

The problem here is that HTTPD knows everything the client browser knows, so the CGI wrapper calling SESSIOND cannot distinguish between them. To avoid this attack, a secure channel must be established all the way from the client browser to the INSIDE compartment. There must be a secret known only to the client, or shared between the client and software running at SYSTEM INSIDE. Also all requests must be signed to make sure they have not been modified by HTTPD. So even if the OUTSIDE compartment or software

running within it were interfered with, no modification or forgery of requests could be accomplished. This is not possible with a vanilla Web browser and HTML. To achieve this we would have to build a client plug-in or investigate use of Java.

V. Future work

V.A Accessing internal systems, multiple independent applications

The principles described above can be applied to build more complex configurations, where the CMW platform is used to run multiple independent applications, and applications which directly access an organization's internal legacy systems.

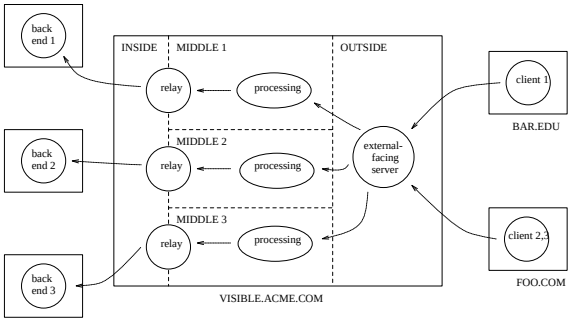

Figure 5: Accessing internal systems

Figure 5 shows this type of configuration. By running each application in a separate compartment, it is possible to ensure that even if one application is compromised, it cannot be used to attack or interfere with the other applications running on the same machine. Where an application has to access systems on the internal LAN, it is not given direct access, but has to talk to them through a trusted relay. The relay can limit which internal machines the application communicates with, and can contain monitoring and logging facilities that ensure only appropriate requests are made, and that check for suspicious activity. This might be particularly attractive to an organization's security audit department, who would gain concrete assurance of behaviour to complement the detailed reviews needed for such applications.

V.B Management tools

On configurations running multiple applications, there will be a need for tools which allow one person to administer specific applications only, and another to manage the underlying system. Practical administration tools must include privileged software that overrides mandatory access controls, without turning into sources of vulnerability. They must not be too big and unwieldy, and must adhere to principle of least privilege. This can be accomplished by defining new roles as necessary and associating appropriate command authorizations with them. In comparison with traditional firewall solutions, the organization will have a smaller number of machines to administer, and will have access to them from the internal network, and will gain the security advantages of having fewer different configurations to maintain in a secure state.

VI. Conclusion

Our investigations have shown that CMW technology is a suitably secure platform for hosting applications which a company wants to be accessible from the external Internet yet which also communicate with the company's internal machines. The mandatory access controls provide boundaries enforced by the kernel to protect systems and services from attack and sensitive information from disclosure; the principle of least privilege allows us to concentrate the functionality which crosses our firewall in a small number of trusted programs.

We have found it relatively straightforward to get new and existing applications to run under the CMW operating system. It appears, in general, that in situations where previously two machines were needed to create an effective security barrier, we can use CMW technology to reduce the need to that of a single machine. The advantages of this are twofold. Firstly, where network performance is an issue, a single CMW machine forms less of a bottleneck. Secondly, the burden of system administration is reduced by having only a single machine.

The services described above have been implemented in an experimental form at Hewlett-Packard Laboratories, Bristol as part of the E2S European collaborative. They have been released to the CMW product division of Hewlett-Packard for consideration as included features of a future version of their products.

Our work on investigating the use of CMW technology continues. Areas of current interest include running multiple applications on a single CMW machine, controlled access to internal legacy software, and, especially, management tools for the CMW platform.

VII. References

[1] Millen, J.K. and Bodeau, D.J., “A Dual-Label Model for the Compartmented Mode Workstation”, MITRE Paper M90-51, The MITRE Corporation, 1990.

[2] National Computer Security Center, “Department of Defense Trusted Computer System Evaluation Criteria”, DOD Standard 5200.28-STD, 1985.

[3] Defense Intelligence Agency, “Compartmented Mode Workstation Evaluation Criteria”, Report DDS-2600-6243-91, 1991.

[4] Hewlett-Packard Co., “HP-UX CMW MaxSix Administrator's Guide”, 1995.

[5] Mockapetris, P.V., “Domain Names: Concepts and Facilities”, RFC 1034, 1987.

[6] Mockapetris, P.V., “Domain Names: Implementation and Specification”, RFC 1035, 1987.

[7] Vixie, P., “DNS and BIND Security Issues”, 5th Usenix Security Symposium, 1995.

[8] Schuba, C.L. and Spafford, E.H., “Addressing Weaknesses in the Domain Name System Protocol”, submitted to the twenty-second Telecomunications Policy Research Conference, 1994.

[9] Chapman, D.B. and Zwicky, E.D., “Building Internet Firewalls” pp. 278-296, O'Reilly and Associates, Inc., 1995.

[10] Cheswick, W.R. and Bellovin, S.M., “Firewalls and Internet Security”, pp. 137-208, Addison Wesley, 1994.

[11] Garfinkel, S. and Spafford G., “Practical Unix and Internet Security”, 2nd Edition, pp. 505-506, O'Reilly and Associates, Inc., 1996

[12] Hewlett-Packard Co., “Virtual Vault Transaction Server Concepts Guide”, 1996.

[13] Netscape Communications Corporation, Persistent Client State - HTTP Cookies, *http://home.netscape.com/newsref/std/cookie_spec.html*, 1997.

[14] Freier, A.O., Karlton, P. and Kocher, P.C., “The SSL Protocol, Version 3.0”, *http://home.netscape.com/eng/ssl3/ssl-toc.html*, 1996.

Author Information

Jonathan Griffin joined Hewlett-Packard Labs. in 1985 after receiving a Master's degree in Electrical Engineering. Since 1994 he has been working in the area of secure electronic commerce and the World Wide Web. Previously he did research in network management and distributed systems security.

After graduating in Electronic Engineering at Imperial College, London, Christopher Dalton worked as programmer for Alfred McAlpine Plc and Lucas Aerospace Engine Systems before joining the University of Wales, Bangor in 1993 as a system programmer and manager of the central Unix systems. In 1996 he joined Hewlett-Packard Labs. and is currently working in the area of secure electronic commerce.