CGI CONTROL OF REMOTE TELECOMMUNICATION EQUIPMENT

J.C. SIMNER*, S. BECK**, M. WUWER**, T. OSMAN*** and D. AL-DABASS***

** Siemens Communications*
Technology Drive, Beeston,
Nottingham, NG9 1LA.
john.simner@siemens.com

*** Siemens Communications*
Munich
Germany.

**** School of Computing & Mathematics*
The Nottingham Trent University
Nottingham, NG1 4BU.
taha.osman@ntu.ac.uk

Abstract: Cordless handsets allow a user to make and receive calls anywhere within the range of the base stations. The base stations provide the low power cellular radio communications to the cordless handsets. The performance of the cordless equipment should be monitored to ensure that calls are not being lost. Users may be aware of some lost calls because they were talking at the time the call failed. They would not be aware of any incoming calls that fail to ring their handsets. Any lost calls could result in loss of business. This paper highlights the limitations of local monitoring. It explores some examples of remote monitoring connected with network management and rail transportation to see whether the technologies used can enhance the collection of cordless statistics on Hicom. It successfully combines technologies from the different examples to control the collection of cordless statistics on the remote telecommunication equipment. It uses a well-established client-server technology in a new way. It does not use it in the normal way to return information in a web page. Instead, it uses it to control the starting and stopping of the statistics collection.

Keywords: network management, remote monitoring

1. INTRODUCTION

Siemens Information and Communication Networks designs, develops, manufactures, and markets telecommunication equipment (termed "switch") that supports cordless equipment. Figure 1 shows a typical Hicom switch with private cordless equipment, telephones, and a local administration and service terminal connected to the Public Network.

Cordless handsets allow a user to make and receive calls anywhere within the range of the base stations. The base stations provide the low power cellular radio communications to the cordless handsets. The performance of the cordless equipment should be monitored to ensure that calls are not being lost. Users may be aware of some lost calls because they were talking at the time the call failed. They would not be aware of any incoming calls that fail to ring their handsets. Any lost calls could result in loss of business.

The goal of this paper is to investigate alternative monitoring scenarios and subsequently, produce a control mechanism that starts and stops the monitoring, which can be applied to a commercial product.

Section 2 highlights the limitations of local monitoring and investigates remote monitoring as an alternative to local monitoring. It explores two examples of remote monitoring (network management and rail transportation) to see whether the technologies used can be applied to cordless.

Section 3 proposes remote collection of the statistics using a well-established client-server technology. It focuses on the control aspects of the remote collection system and shows how the chosen technology can be used to convey the user's requests between different parts of the system. To protect customers' investment, the chosen technology must run on both old and new Hicom switches. This limits the choice to those technologies that are supported on old switches.

Section 4 surveys different variants of the client-server technology and considers how appropriate they are to the proposed solution.

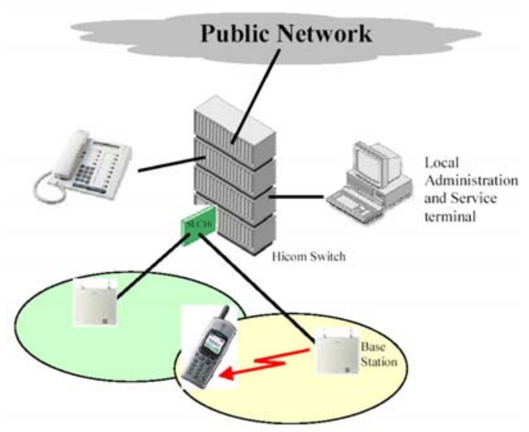

Figure 1 - Typical Private Cordless Telecommunication Equipment

Section 5 outlines tests performed and tools used to gauge how successful the chosen client-server technology is in meeting its requirements. It identifies security issues with the chosen technology and proposes using a more secure form for telecommunication equipment.

2. LIMITATIONS OF LOCAL MONITORING

Section 1 highlighted the importance of monitoring the cordless equipment to ensure that it provides the best service and that cordless calls are not lost.

The switch can measure and record many different aspects of cordless calls. It records the number of times a particular aspect has occurred since the switch was last reset. This provides an absolute measurement rather than a historical record over time.

The recorded information is accessed using a proprietary management interface. An engineer can generate historical information by manually invoking the collection commands on a periodic basis, collecting their output, extrapolating the current values, and comparing them with the previous values. Even on a small switch, the collection commands can generate hundreds or thousands of lines of textual output.

There are two ways of accessing the management interface; locally or remotely through a modem. The main problem with the remote connection is the speed of the modem link especially with the amount of textual output generated. Sometimes, the periodic period is increased because of the time it takes to receive the generated output. This could affect the worth of the historical data.

Hence, when monitoring produces a large amount of data, the most efficient way to collect it is to send an engineer to site with a laptop to locally collect the statistics. This process is very expensive in time and manpower.

The statistics are collected in blocks of 15 minutes. Whilst, the statistics are being collected, the engineer must remain on site. A typical site visit to collect the statistics is 3 hours plus travel time.

However, when monitoring produces a small amount of data, it can be remotely collected through a modem.

ALTERNATIVE MONITORING APPROACH

The previous section highlighted the cost of local monitoring. This section investigates remote monitoring as an alternative to local monitoring. There are many published examples of remote monitoring. This section considers two examples; network management and rail transportation.

Remote Monitoring Example - Network Management

Network Management Systems are a typical example of remote monitoring. Typically, a central network manager polls the nodes, collects data from them, processes it, and presents it in a visual form to a human operator.

Kooijman (1995) and Gavalas et al. (2000) propose using agents to reduce the amount of data passed between the nodes and the server and the amount of processing done by the server.

The current cordless monitoring approach is inefficient because it transfers so much data.

Network management and agent technology have not been explored further because agent technology is not supported on old switches.

Remote Monitoring Example – Rail Transportation

Nieva, Fabri, and Wegmann (2001) and Fabri, Nieva, and Umiliacchi (1999) developed " a web-based monitoring tool for trains … [that] allow[ed] maintenance staff to supervise railway equipment from anywhere at anytime." (Nieva, Fabri, and Wegmann 2001, p1)

They identified the significant benefits including; reduced development, installation, and maintenance personal travel costs. These cost savings are just as pertinent for a service organisation.

They developed and compared three prototypes based upon different technologies; HTTP with CGI, Java RMI, and HTTP with XML. The prototypes were used to monitor a single device on a train, all devices on a single train, and all devices on a fleet of trains, respectively.

They found that the CGI approach was a "fast-to-develop and elegant [solution]" (Fabri, Nieva, and Umiliacchi 1999, p12) that suffered from using a proprietary protocol between the client and the server.

The Java RMI approach pushed the data from the server whilst both HTTP approaches pulled it. Nieva, Fabri, and Wegmann found firewall security problems with pushing the data. The main advantage of pushing over pulling is the reduction in communication overhead because the data is only sent when it changes.

The XML approach enabled the data and its meaning to be sent to the client. This allows the data to be interpreted by the client.

Nieva, Fabri, and Wegmann compared the performance of the three prototypes for one and ten updates. They found that the HTTP approach is slower than the Java RMI approach, and "the difference between the performances of Java RMI against HTTP will increase as we increase the number of updates." (Nieva, Fabri, and Wegmann 2001, p5).

The two HTTP approaches, HTTP with CGI and HTTP with XML, could both be used to collect the cordless statistics. The CGI approach is quicker to develop than the XML approach but the XML approach would allow for future enhancements as the data and its meaning are both collected. However, the CGI approach operates faster than the XML approach as less data is being transferred between the switch and the control centre.

The Java RMI approach is not appropriate to cordless because it uses Java technology on both the client and the server. The Java technology could be added to new switches but is not supported on old switches.

3. THEORY OF NEW IDEA

The previous sections highlighted limitations with local monitoring and investigated remote monitoring as an alternative. This section proposes a remote collection system for Hicom using a client-server technology.

There are a number of aspects to the remote collection system; controlling the starting and stopping of the collection process, running the collection commands, processing and transferring the collected data. Section 3.1 describes the remote cordless collection scenario that the new idea must work within.

The remainder of this paper focuses on the control aspect between the Manager and the Hicom switch using a client-server technology. It does not address the collection process, which uses an established mechanism.

3.1 Remote Cordless Collection Scenario

Figure 2 shows the remote cordless collection scenario. There are three areas; the Administration and Service (A&S) Client, the A&S Platform on an Intel-based Server (termed "the Manager"), and the A&S Platform on a proprietary card (ADP) installed within the Hicom switch (termed "the Assistant"). The Manager can remotely access one or more Assistants.

A user logs onto the Manager to control and view the statistics on the Hicom switch. When the user starts or stops the data collection, the request is conveyed through the Manager to the Assistant. The Assistant periodically collects the data from the Hicom switch by invoking the collection commands and analysing their output. Subsequently, the Manager remotely collects the processed data from the Assistant.

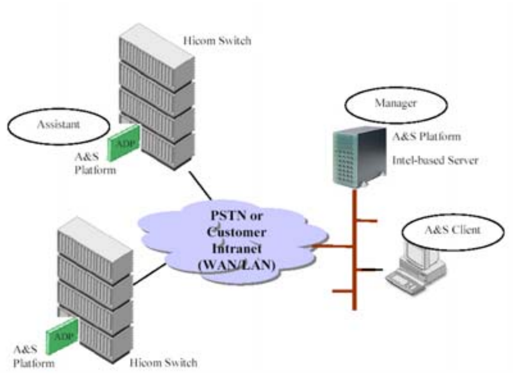

Figure 2 - Remote Cordless Collection Scenario

Overview of Control using Client-Server Technology

The control aspects of the remote collection system are the passing of the user's requests (i.e. start collection and stop collection) from the Manager to the Assistant. To protect customers' investment, the chosen client-server technology must run on both old and new Hicom switches. This limits the choice to those technologies that are supported on old switches. This means that it must be a well-established technology rather than one developed in the last few years.

The Apache web server and Common Gateway Interface (CGI) were chosen. They are freely available, UNIX-based, non-proprietary and widely used.

CGI is used to convey the user's requests to the Assistant. Normally, CGI is used to return a web page to the client but the remote cordless collection system uses it to control the collections. The alternative CGI invocations that can be used to convey these requests are described later.

Using CGI allows remote access through the Internet, Intranet, or any other open network to the telecommunication equipment. There is concern that such access will allow the telecommunication equipment to be more open to attack. As it is impossible to eliminate these attacks, their effects must be minimised.

Whilst, the CGI script is being invoked, there is very limited feedback. It does not return success or failure. This minimises the information returned to a potential hacker. In addition, a barrier is required between the CGI script interface and the rest of the collection process. The barrier must utilise minimum resources; memory and processor. A natural barrier is the creation and deletion of a control file. To ensure that the barrier is effective, the CGI script interface must not allow; any user input to be executed by the system or the user to interrupt it and take control.

Figure 3 shows an overview of the remote collection system focusing on the control aspects.

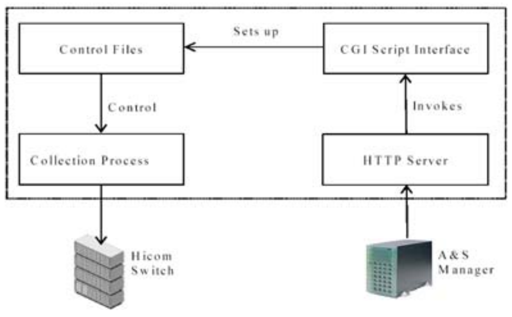

Figure 3 - Remote Collection Overview – Control Aspects

CGI Invocations

CGI is used to convey the user's request from the Manager to the Assistant. Figure 4 shows two alternative invocations.

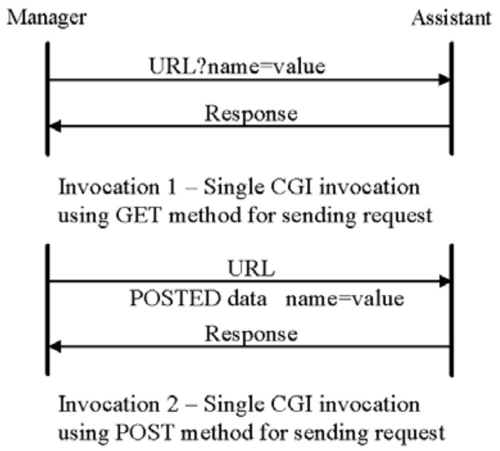
Manager
Assistant
URL?name=value
Response
Invocation 1 – Single CGI invocation using GET method for sending request
URL
POSTED data name=value
Response
Invocation 2 – Single CGI invocation using POST method for sending request

Figure 4 - Alternative CGI Invocations

The simplest CGI invocation is the GET method. The user's request is appended to the URL. The GET method is very insecure. The URL and the user's request can appear in the browser location bar and be logged by any system the request travels through. With no GUI, there is no browser location bar so the only concern is the logging by other systems.

An alternative CGI invocation is the POST method. The user's request is transmitted immediately after the URL. One advantage of this method is the unseen data. The Manager requires unseen data to hide the security measures.

Kargl, Maier, and Weber (2001) identify that a system can be attacked at different levels. This section focuses on the user's request protocol between the Manager and the Assistant. Eronen (2001) and Moore, Voelker, and Savage (2001) identify the difficulties in detecting and tracing flood attacks, which consume CPU and memory resources. Eronen and Meadows (2000) identify countermeasures to distinguish the valid requests from the rogue requests.

As the GET and POST invocations have no protection against denial of service attacks, security measures or a commercial product (e.g. Password Hurler Protection) are required to reduce the effect of flood attacks.

The security measures enable the Assistant to validate the user request with minimum processing and memory utilisation. It can check IP address of the visitor, the content length, and key value pairs. If the request is invalid, it is immediately ignored and nothing is returned. This behaviour is loosely based on Gong and Syverson's (1995) fail-stop protocol.

Gong and Syverson state that "A fail-stop protocol automatically halts when there is any derivation from the designed protocol execution path." (Gong and Syverson 1995, p2). The user's request protocol conforms to Gong and Syverson's Definition 1 (Fail-Stop Protocol) (Gong and Syverson 1995, p3) because it returns nothing if the request is invalid. However, this is the only conformance because it uses weak not strong authentication.

Eronen and Moore, Voelker, and Savage report that attackers will often forge or "spoof" the source IP address so that they can not be traced. Therefore, the source IP address must be checked.

Three different levels of checks can be performed:

1. Does the source IP address exist in the HTTP request? Some surfers forcibly remove their address from their request. As the user's request protocol always sends the address, any HTTP request received without an address must be an attack.

2. Is the IP address valid? It is very difficult for the recipient to check the validity of an IP address. It can check it is in the right range but it can not check that it equates to a valid location. The attacker probably selected the address at random. The location may not exist or it may be the address

of an innocent third party. Hence, this check is fallible and should not be used.

3. Is the IP address trusted? The Assistant could hold a list of the Manager IP addresses. Whenever, it received an HTTP request, it could validate the source IP address received against the list of Manager IP addresses. If there was no match, the HTTP request must be an attack. Unfortunately, it could also mean that a valid request was received from a Manager but the list has not been updated yet. The validity checks consume time, CPU, and memory resources. The amount of resources used depends on the number of Managers and the position of the received address in the list. Therefore, this check should be used as a last resort.

Meadows and Eronen identify that alternative techniques are required "to prevent attacks which employ IP spoofing." (Eronen 2001, p4). Meadows proposes authentication whilst Eronen proposes cookies.

The HTTP request could contain an additional key value pair, which is authenticated by the Assistant. If the Assistant detected an invalid key value pair, the request must be an attack. This approach is not appropriate to the GET method because the security measure could be logged and sent in subsequent attack requests.

The website for the Password Hurler Protection (www.passwordhurlerprotection.com) states that "[it] stops brute force attacks on your web site… [it] works by logging the IP address of all failed logins (401 Errors) and then it blocks users based upon the number of failed logins within a specific period of time."

This level of protection is valid for many commercial web sites and may be appropriate for telecommunication equipment. In the case of the Assistant, the alternative approach of validating known Manager IP addresses and blocking all other addresses should consume fewer resources than logging and blocking failed logins especially if the failed logins are mounting a distributed denial of service attack.

4. DEVELOPMENT OF CGI SCRIPT INTERFACE

There are a number of alternative CGI approaches. This section provides a comparative survey of the different approaches and considers how appropriate they are to the proposed solution.

Shah and Darugar (1998), Venkitachalam and Chiueh (1999), Wu, Wang, and Wilkins (2000) and Dumitrescu (1998) all identify that CGI has an inherent performance problem because separate processes are created to handle each client request. They all highlight the overheads incurred in forking a new process.

New approaches have been developed to overcome the performance problems. Some are proprietary Server APIs (e.g., mod_perl) whilst others are modifications to the CGI execution architecture (e.g., FastCGI, LibCGI, and VEP).

Mod_perl (http://perl.apache.org/guide) brings together the PERL application and the Apache web server into one process.

FastCGI is described by Venkitachalam and Chiueh and on the FastCGI web site (http://www.fastcgi.com, Brown 1996a, Brown 1996b & Open Market 1996). It runs as a persistent process thereby eliminating the overheads of creating a new process.

Venkitachalam and Chiueh advocate a high-performance CGI architecture, LibCGI. The CGI script is compiled into a shared library that executes in the web server's address space. It avoids the overhead of executing the forked process.

Shah and Darugar cite a high performance architecture using Binary Evolution's VelociGen™ interface (see Shah and Darugar 1998, p2). They describe VelociGenforPerl™ (VEP) which "combines the performance associated with server APIs with the benefits of CGI." (Shah and Darygar 1998, p2).

Table 1 shows a comparative summary analysis of the five approaches. The information has been extracted from the referenced papers.

Attributes	**CGI**	**Mod_perl**	**FastCGI**	**libCGI**	**VEP**	**Server API**
Performance	Poor	Fast	Fast	Fast	Fast	Fast
Separate Isolated Process	Yes	No	Yes	No	Yes	No
Persistent Process	No	Yes	Yes	No (Shared library)	Yes	Yes
Language Independence	Yes (Perl, C, C++)	No (Only Perl)	Yes	No (Only C, C++)	No (Only Perl)	No (Usually C, C++)
Proprietary	No	Yes	No	Yes	Yes	Yes
Architecture Independence	Yes	No	Yes	No	Yes	No (Same as web server)
Ease of Use	Simple	Simple (Can run existing Perl scripts)	Simple (Easy migration from CGI)	Simple (Similar to conventio-nal CGI)	Simple (Can run existing Perl scripts)	Usually Complex

Table 1 - Comparative Summary Analysis of CGI Approaches

Two of the important attributes in the above table which significantly effect performance and security are separate isolated process and persistent process.

With a separate isolated process, a CGI based application crash will not bring down the entire web server. If they shared the same process space, the application can corrupt, crash, or compromise the web server. The application could even access the session keys for the encryption. Venkitachalam and Chiueh describe LibCGIs solution to sharing the same process space so that the application does not corrupt, crash, or compromise the web server.

Persistent processes do not die when they have finished handling a request. Instead, they wait around for a new request.

Venkitachalam and Chiueh compare LibCGI with two alternative solutions, FastCGI, and mod_perl. They conclude "LibCGI improves the CGI script execution throughput over FastCGI by a factor of 2.3, and over conventional CGI model by a factor of 3.9 to 4.6." They compared performance at the machine level and across the network.

Kothari and Claypool (1999) also measured and analysed the performance of CGI and FastCGI for input data size, output data size, disk read, disk write, and computation. They found that "CGI and Fast CGI perform effectively the same under most low-level benchmarks."

A Technical White Paper on FastCGI (Open Market 1996) compared FastCGI with CGI and concluded that FastCGI was 5 times faster than CGI.

Shah and Darugar compared VEP with CGI and concluded that VEP was up to 20 times faster than CGI.

In contrast, Wu, Wang, and Wilkins conclude that "CGI solutions are appropriate for small applications with a limited amount of client access … with the trade-off being the performance penalty." (Wu, Wang, and Wilkins 2000, p10)

This implies that CGI can be used to control remote monitoring (i.e. starting and stopping) because the requests occur very infrequently. Typically, there will be one request to start the monitoring and another some time later to stop it.

The High Performance Common Gateway Interface Invocation paper by Venkitachalam and Chiueh covering CGI performance problems, LibCGI, FastCGI, and mod_perl were evaluated.

The conclusions were;

- There are recognised performance problems with CGI,

- New approaches have been developed that overcome these problems,

- The new approaches are not appropriate to the control of remote collection of cordless statistics. As there are minimal requests between the Manager and the Assistant, any performance problems with CGI are not seen as an issue.

- Therefore, CGI will be used to convey the user's request from the Manager to the Assistant.

The CGI script interface only runs on the ADP (see Figure 2). The operating system on the ADP is UnixWare. The UnixWare operating system provides a comprehensive environment with many shells, commands, functions, and tools.

The environment influences the form (e.g. executable or shell script) and choice of the programming language (e.g. C, C++, or Perl) for the CGI program.

An important aspect of the control using client-server technology proposal is the barrier between the CGI script interface and the rest of the remote collection system. The barrier is achieved through the creation and deletion of a control file.

Two Bourne shell scripts are used; one to create the control file and the other to delete it. The Bourne shell scripts were used in preference to an executable because when they are called, a new process is not created so there are no additional overheads.

This leaves the choice of the programming language for the CGI program.

Gundavaram states that "Perl is by far the most widely used language for CGI programming!" (1996, p11). He cites one of the advantages of Perl as "It makes calling shell commands very easy, and provides some useful equivalents of certain UNIX system functions." (1996, p11).

With this recommendation and the two Bourne shell scripts already developed, Perl was the obvious choice for the CGI program especially as Gundavaram (1996, p65) has a standard CGI PERL script that can be used. The only additions required are the recognition of the key-value pairs and the calling of the Bourne shell scripts.

5. RESULTS AND DISCUSSION

With the controlling of the remote collection system, two separate functional tests are required to test the CGI script interface; one to start the collection and the other to stop it. In both cases, the URL is invoked and the correct operation was tested and checked. The start request created the control file whilst, the stop request deleted it.

The tests were successful. They clearly demonstrated that the cordless collection could be remotely started and stopped across the Siemens network. The switch was located in the lab and the testing was carried out from a PC in the office.

The CGI invocations using the GET method were easily tested using the Internet Explorer browser to invoke the URLs from the address line. As this approach does not work for the POST method, alternative approaches were investigated. A free command line tool, cURL (http://curl.haxx.se), was found which can transfer files with URL syntax. It supports many different aspects of client-server technology. It was successfully used to test the normal and error behaviour of the CGI script interface.

Although the tests were successful, they did identify security issues with the simple CGI script interface. It minimised attacks but did not prevent unauthorised access. Hence, a more secure CGI invocation is required for telecommunication equipment.

The following paragraphs outline the principles of secure CGI within the context of conveying user's requests from the Manager to the Assistant.

First, all communication between the Manager and the Assistant uses HTTPS (HTTP over SSL) rather than the standard HTTP. Therefore, all information exchanged between the Manager and the Assistant is encrypted. This includes the header, URL, posted data, and any cookies. The name of the server is not encrypted because it is used to route the request. Encryption does not stop any system the request travels through from seeing the information; it just makes it difficult for them to decode. With HTTPS and no GUI, the security measures can be placed in the URL and/or the posted data.

Secondly, authentication and/or cookies are required to distinguish the authorised accesses from the unauthorised ones.

With secure CGI invocation, the user is expected to login before the URL is invoked. If the user attempts to invoke a URL before they have logged in, they are automatically redirected to a login page. When they have successfully logged in, the original URL is automatically invoked. Therefore, the Assistant must be able to determine if the user is already logged in.

There are two possibilities; the user name is sent in every request and the server checks that the particular user has already logged in, or a cookie is sent in every request after the user has logged in. As the initial request has no cookie, the automatic redirection to the login page occurs. When the user has successfully logged in, the server puts a cookie onto the client, which is returned in subsequent requests.

As the cookie is a simpler and more efficient approach than searching for logged on user names, they are used in this invocation.

Unfortunately, with the user's requests, there is no browser or logged on user, there is only an executable running on the Manager invoking a URL on the Assistant. The executable could detect the login page and login but the user name and password would have to be hard coded or easily available. This presents a number of security problems.

For example, hard coded passwords can be easily detected and difficult to change. As the password should be changed on a regular basis to avoid misuse, hard coded passwords should not be used.

Therefore, an alternative approach to user name and password is required.

The Photuris Specification (RFC 2522) outlines some basic requirements for cookie generation. They include:

"1. The cookie MUST depend on the specific parties …

2. It MUST NOT be possible for anyone other than the issuing entity to generate cookies that will be accepted by that entity. This implies that the issuing entity will use local secret information in the generation and subsequent verification of a cookie. …

3. The cookie generation and verification methods MUST be fast to thwart attacks … " (Karn and Simpson 1999, p19).

These requirements can be adapted to allow a Manager to open a session to the Assistant. Figure 5 shows a secure CGI invocation between the Manager and the Assistant.

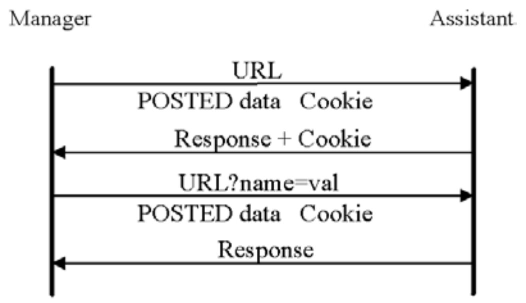

Figure 5 - Secure CGI Invocation between Manager and Assistant

The initial request can include a cookie, which identifies the Manager, the Assistant, and type of request. The Assistant can use these details to verify that the request has come from a Manager. Subsequently, the Assistant can return a cookie, which is sent in subsequent requests from the Manager.

Therefore, Managers and Assistants can easily distinguish between valid requests and responses, and can determine unauthorised access and potential attacks by the existence or not of valid cookies. If an invalid cookie is detected, it must be an unauthorised access or attack and the request is simply ignored.

The tests were successfully repeated using the secure CGI invocations. The Manager detected and ignored unauthorised accesses.

6. CONCLUSIONS

This paper highlighted limitations with local monitoring and found that remote monitoring was a viable alternative.

This paper explored two examples of remote monitoring; network management and rail transportation. These examples were chosen because they are monitoring real time systems, which have similar characteristics to telecommunication equipment. The examples identified a number of underlying technologies that could be used for the collection of cordless statistics; agents, HTTP with CGI, Java RMI, and HTTP with XML.

To protect customers' investment, the remote collection solution had to work on both old and new Hicom switches. Some technologies (e.g. agents and Java RMI) had to be discarded because they were not available on old switches. Other technologies (e.g., HTTP with XML) were not suitable because of large memory footprints or performance problems.

The chosen underlying technology was HTTP with CGI. The remote collection solution did not use CGI in the normal way to return the collected statistics in a web page. Instead, it used it to control the starting and stopping of the cordless statistics. No feedback was given in order to confuse any potential hackers.

This paper investigated the performance problems of CGI including surveying alternative CGI. It concluded that any performance problems were not an issue because there are minimal CGI requests when CGI is used to control.

This paper recognised that CGI is not a secure technology. It investigated alternative CGI invocations with different security measures that included verifying the source IP address, using HTTPS, adding key value pairs and cookies to distinguish between valid and invalid requests between Managers and Assistants.

Finally, the goals of this paper were met as the proposed solution was adopted in the Siemens HiPath 4000 Administration and Service Product to control the collection of the cordless statistics from switches.

ABBREVIATIONS

The following abbreviations have been used in this paper:

A&S -	Administration and Service
ADP -	Administration Data Processor
CGI -	Common Gateway Interface
CPU -	Central Processing Unit
GUI -	Graphical User Interface
HTTP -	HyperText Transfer Protocol
HTTPS -	HyperText Transfer Protocol Secure
ICN -	Information and Communication Networks
IP -	Internet Protocol
PSTN -	Public Switch Telecommunication Network
RFC -	Request For Comment
RMI -	Remote Method Invocation
SLC -	Subscriber Line Cordless
SSL -	Secure Sockets Layer
URL -	Uniform Resource Locator
XML -	Extensible Markup Language

ACKNOWLEDGMENTS

I would like to thank Siemens ICN and especially my immediate management; Mr. Roger Andrews, Mr. Paul Erckens, Mr. Jeff Conway, and Mr. Graham Underwood for giving me the opportunity, time, and support to do the MSc and this paper.

Finally, I owe a lot of gratitude and thanks to my wife Mrs. Janet Simner and children David and Andrew for their love, support, and encouragement whilst doing this paper.

AUTHOR

John Simner is a senior software engineer in Siemens' Design Services at Nottingham, U.K.. He graduated from the University of Birmingham in 1978 with a BSc with Honours Class I in Electronic and Electrical Engineering. He has worked in the Telecommunication Industry for over 25 years, working on real-time embedded and application software in C, C++, and Java. Currently, he is part of a team enhancing a web-based administration and service (A&S) product developed by Siemens Information and Communication Networks. He was part of the first cohort on a MSc course set up between NTU and Roger Andrews Siemens' Head of Engineering. This paper is taken from his MSc. project which developed an application that remotely collected Cordless Telecommunication statistics from HiPath 4000 telecommunication equipment.

REFERENCES

BROWN, M.R., 1996a. **FastCGI: A High-Performance Gateway Interface**. Open Market, Inc. <http://www.fastcgi.com.devkit/doc/www5-api-workshop.htm> (3 July 2002)

BROWN, M.R., 1996b. **Understanding FastCGI Application Performance**. Open Market, Inc. <http://www.fastcgi.com.devkit/doc/fcgi-perf.htm> (3 July 2002)

DUMITRESCU, R.A., 1998. **Two-stage Programming via the Client-Servlet-Coprocess Interaction Model**. University of Basel, Switzerland.
(Source http://citeseer.nj.nec.com/77511.html Cached: PDF, 12 June 2002)

ERONEN, P., 2001. **Denial of service in public key protocols**.
Helsinki University of Technology.
(Source http://citeseer.nj.nec.com/eronen01denial.html Cached: PDF, 11 May 2002)

FABRI, A., NIEVA, T. & UMILIACCHI, P., 1999. **Use of the Internet for Remote Train Monitoring and Control: the ROSIN Project**.
Paper appeared in the Proceedings of Rail Technology '99, London, September 1999.
(Available http://icawww.epfl.ch/nieva/thesis/Conferences/RailTech99/article/RailTech99.pdf)

GAVALAS, D., GREENWOOD, D., GHANBARI, M. & O'MAHONY, M., 2000. **Advanced Network Monitoring Applications Based on Mobile/Intelligent Agent Technology.**
University of Essex, Colchester, UK & Fujitsu Telecommunications Europe Ltd., UK.
(Source http://citeseer.nj.nec.com/268291.html Cached: PDF, 18 July 2002)

GONG, L. & SYVERSON, P., 1995. **Fail-Stop Protocols: An Approach to Designing Secure Protocols.** SRI international, Menlo Park, California.
Paper to appear in Proceedings of IFIP DCCA-5, Illinois, September 1995.
(Source http://citeseer.nj.nec.com/49099.html Cached: PDF, 11 May 2002)

GUNDAVARAM, S., 1996. **CGI Programming on the World Wide Web**. 1st ed.
Sebastopol, CA: O'Reilly & Associates, Inc.

KARGL, F., MAIER, J. & WEBER, M., 2001. **Protecting Web Servers from Distributed Denial of Service Attacks.** University of Ulm, Germany.
(Source http://citeseer.nj.nec.com/444367.html Cached: PDF, 11 May 2002)

KARN, P. & SIMPSON, W., 1999. **Photuris: Session-key Management Protocol.**
Network Working Group, Request for Comments 2522 (RFC 2522), Category: Experimental.
(Source http://rfc.sunsite.dk/rfc/rfc2522.html, 8 September 2002)

KOOIJMAN, R., 1995. **Divide and conquer in network management using event-driven network area agents.**
(Source http://citeseer.nj.nec.com/Kooijman95divide.html Cached: PDF, 18 July 2002)

KOTHARI, B. & CLAYPOOL, M., 1999. **Performance Analysis of Dynamic Web Page Generation Technologies.** Computer Science Technical Report Series. WPI-CS-TR-99-12
Worcester Polytechnic Institute, Massachusetts.
(Source http://citeseer.nj.nec.com/119628.html Cached: PDF, 12 June 2002)

MEADOWS, C., 2000a. **A Cost-Based Framework for Analysis of Denial of Service in Networks.** Naval Research Laboratory, Washington, DC 20375.
(Source http://citeseer.nj.nec.com/375643.html Cached: PDF, 11 May 2002)

MEADOWS, C., 2000b. **A Framework for Denial of Service Analysis.**
Naval Research Laboratory, Washington, DC 20375.
(Source http://citeseer.nj.nec.com/484887.html Cached: PDF, 11 May 2002)

mod_perl guide.
<http://perl.apache.org/guide/intro.htm> (3 July 2002)

MOORE, D., VOELKER, G.M. & SAVAGE, S. 2001. **Inferring Internet Denial-of-Service Activity.** University of California, San Diego. (Source http://citeseer.nj.nec.com/moore01inferring.html Cached: PDF, 11 May 2002)

NIEVA, T., FABRI, A. & WEGMANN, A., 2001. **Remote Monitoring of Railway Equipment using Internet Technologies.** Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland & ABB Corporate Research Ltd., Baden, Switzerland. (Available http://icawww.epfl.ch/nieva/thesis/TechnicalReports/RMREIT/TR01_018.pdf)

Open Market, Inc., 1996, Technical White Paper. **Fast CGI: A High-Performance Web Server Interface.**
<http://www.fastcgi.com.devkit/doc/fastcgi-whitepaper/fastcgi.htm>
(3 July 2002)

SHAH, A. & DARGAR T., 1998. **Creating High Performance Web Applications Using Perl, Display Templates, XML, and Database Content.** Binary Evolution, Inc.
(Source http://citeseer.nj.nec.com/112243.html Cached: PDF, 11 June 2002)

VENKITACHALAM, G. & CHIUEH, T., 1999. **High Performance Common Gateway Interface Invocation.** State University of New York at Stony Brook, Stony Brook, NY.
(Source http://citeseer.nj.nec.com/77638.html Cached: PDF, 11 June 2002)

WU, A.W., WANG, H. & WILKINS, D., 2000. **Performance Comparison of Alternative Solutions For Web-To-Database Applications.** Proceedings of the Southern Conference on Computing. The University of Southern Mississippi, October 26-28, 2000.
(Source http://citeseer.nj.nec.com/428587.html Cached: PDF, 11 June 2002)