Identifying Critical Components During Information Security Evaluations

Andrew Rae

Invensys Rail Systems Australia
Brisbane Technology Park
Queensland 4113 Australia
Andrew.J.Rae@invensys.com

Colin Fidge

Queensland University of Technology
School of Software Engineering and Data Communications
Queensland 4001 Australia
c.fidge@qut.edu.au

Electronic communications devices intended for government or military applications must be rigorously evaluated to ensure that they maintain data confidentiality. High-grade information security evaluations require a detailed analysis of the device's design, to determine how it achieves necessary security functions. In practice, such evaluations are labour-intensive and costly, so there is a strong incentive to find ways to make the process more efficient. In this paper we show how well-known concepts from graph theory can be applied to a device's design to optimise information security evaluations. In particular, we use end-to-end graph traversals to eliminate components that do not need to be evaluated at all, and minimal cutsets to identify the smallest group of components that needs to be evaluated in depth.

Keywords: information security, computer communications devices, security evaluations

ACM Classification: K.6.5 (Management of Computing and Information Systems—Security and Protection), C.2.0 (Computer-Communication Networks—General), C.3 (Special-Purpose and Application-Based Systems), J.7 (Computers in Other Systems)

1. INTRODUCTION

Electronic communications devices safeguard classified information in government and military networks. In particular, *domain separation* devices allow the flow of information between high and low-security domains to be controlled. Examples of such devices include data diodes, multi-computer switches, context filters and cryptographic devices.

Before such a device can be deployed, however, it must be carefully evaluated to ensure that it provides the necessary security functionality. We are particularly interested in 'high-grade' information security evaluations, which include a detailed analysis of the device's design.

International standards already exist for information security evaluations. Two especially influential standards are the *Information Technology Security Evaluation Criteria* (Commission of

Manuscript received: 23 August 2004
Communicating Editor: Sidney A. Morris

the EC, 1991) and the more recent *Common Criteria for Information Technology Security Evaluation* (Common Criteria, 1999*a*). However, while these general standards provide helpful frameworks for managing evaluations, their accompanying methodologies (Commission of the EC, 1993; Common Criteria, 1999*b*) do not include specific techniques for undertaking evaluations at high security levels. For instance, the evaluation methodology for the seven-layered Common Criteria standard (Common Criteria, 1999*b*) covers the lowest four security levels only.

Lacking detailed technical guidance, evaluators of information security devices intended for critical applications must therefore take a very conservative approach. In the worst case, this may involve careful scrutiny of *every* component within a device, to determine both how it contributes to the device's security functionality, if at all, and the security consequences of its failure, if any. In practice, such evaluations are time-consuming, labour-intensive and costly.

Our goal in this paper, therefore, is to show how the evaluation process can be made more efficient, by prioritising the components to be studied. In effect, our approach formalises the notion of a 'security perimeter' (Young, 1991). Specifically, we exploit well-known principles from graph theory to:

- eliminate components that do not need to be evaluated at all, because they do not lie on a critical information-flow path; and
- choose components for detailed evaluation, based on whether or not they form crucial links between the high and low-security domains.

The concepts are illustrated through a small case study involving evaluation of a cryptographic device for potential breaches of (overt) data confidentiality.

2. DEVICE MODEL

We assume a simple generic model of the device being evaluated, consisting merely of a graph of *components* linked by *connections*. The level of detail in the model is not significant. It may be a block diagram, with only a few components, or an electronic schematic diagram, with dozens of components. For our purposes we merely view it as a directed graph of nodes and arcs.

The connections may carry, and the components may contain or generate, *information*. Certain components may act as information *sinks*, and others as information *sources*. Typically the aim of a security evaluation is to determine what kinds of information can reach sinks in a 'low-security' domain from sources in a 'high-security' one.

2.1 Information Model

For the purposes of the case study in Section 3 we assume that the information manipulated by components and connections is categorised as follows.

- A *data* connection is one that may carry information that is meaningful in the external security domains. In particular, it may carry *classified* information. Similarly, a 'data' component may contain or generate information that is meaningful externally.
- A *control* connection carries information that is meaningful only to the components within the device being evaluated. Control connections are not usually considered capable of carrying classified information. Similarly for 'control' components.

If a connection or component is capable of manipulating both data and control information, or if we are unsure what kind of information it handles, then we conservatively classify it to be of 'data' type since this is the most worrisome category from a security perspective.

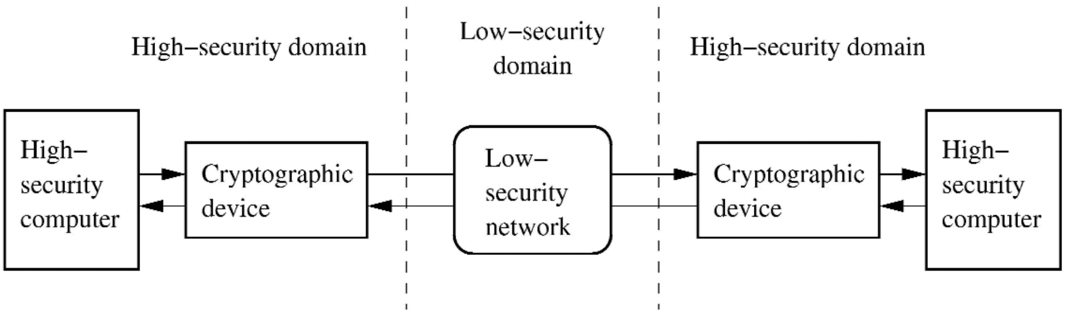

Figure 1: Typical Network Configuration for Cryptographic Devices

The information manipulated by connections and components may also be categorised according to its security level (Committee on NSS, 2003).

- A *red* connection is one that may carry classified information, i.e., information originating from a high-security source. Similarly, a 'red' component is one which may contain or produce classified information.
- A *black* connection carries unclassifed information only, typically originating from the low-security domain. Similarly, 'black' components are those that contain and generate unclassified information only.

Combinations of red/black and data/control information types are possible. Typically, an information security evaluator's primary concern is to trace the flow of *red data* information. (Red *control* information may be of interest during covert channel analyses, as discussed in Section 5.) Black information is usually considered to be of minor interest only.

2.2 Component Model

We also assume that the components within the device can be categorised according to their effect on the flow of information. They can be viewed as functions from the information appearing on their input connections to the information produced on their output connections. Some typical types of component are as follows.

- A *buffer* has an input connection and an output connection. The buffer transfers its input's information to its output with no significant alteration (other than the delay induced by passage through the buffer). Buffers are of relatively little importance during information security evaluations since they allow red data to pass through without hindrance.
- A *switch* typically has an input control connection, and both an input and an output data connection. Information flow from the data input to the data output is determined by the control input. Switches evaluated in isolation are not usually helpful in determining the flow of red data, but they are significant because they may divide the overall security argument into a number of different cases, one for each of the control values.
- A *downgrader* lowers the security significance of its inputs. Typically it will have a red input and a black output. Examples include context filters and encryption components. Downgraders are especially helpful during security evaluations because they may act as sinks for red data, and may thus prevent it from reaching the low-security domain.

- An *upgrader* raises the security significance of its inputs. Typically it will have a black input and a red output. Examples include key and password generators, and decryption components. Upgraders are important for security evaluations because they may act as sources of red data that could eventually reach the low-security domain.

Often it will not be immediately obvious into which of these categories a given component belongs. It is the security evaluator's task to assign a category to each of the significant components, as part of the overall security argument.

3. CASE STUDY

In this section we introduce a particular domain-separation device to serve as an illustrative example for the evaluation techniques presented in Section 4.

3.1 A Cryptographic Device

We consider a prototype 'cryptographic device' (Graves, 2003), which encrypts plaintext characters received from a local computer and sends the resulting encrypted characters to a communications network. It also decrypts encrypted characters received from the network and forwards the resulting plaintext characters to the local computer. The device uses one of several different encryption algorithms, depending on its operating mode. It changes encryption and decryption modes in response to special control characters received from the computer and network, respectively.

In a typical configuration, as shown in Figure 1, such devices are used in pairs to allow high-security computers to communicate over a low-security network. Notice that the cryptographic devices themselves reside in high-security domains.

The particular device of interest is shown in Figure 2. Its rear panel houses a DC power input socket, and two RS232 serial data sockets for connecting to the local computer and the network. The front panel houses a number of switches and indicators, including the power switch and LED.

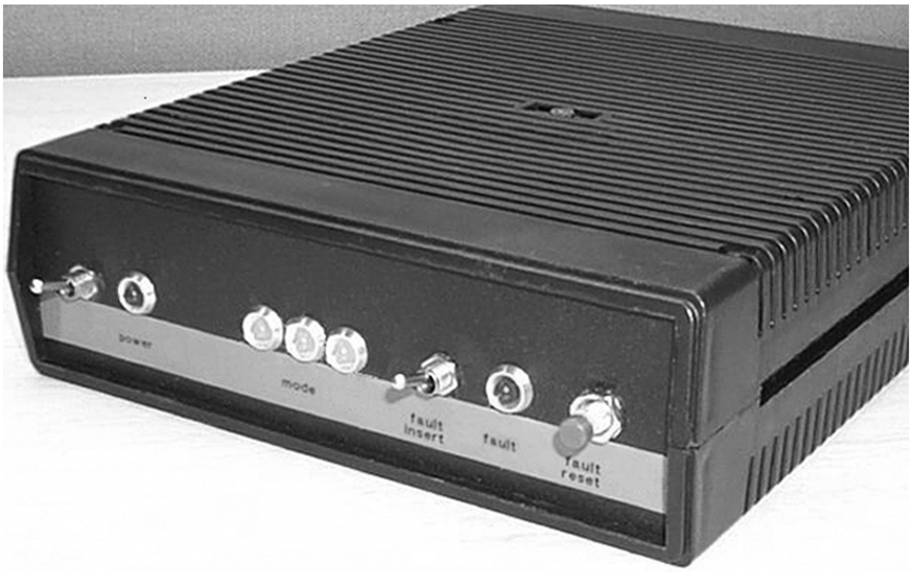

Figure 2: The Cryptographic Device

The device can operate in any of four cryptographic modes, including a non-encrypting 'bypass' mode. The three 'mode' LEDs tell the operator which encryption mode is active, if any.

The device also has an integral self-diagnosis capability, intended to ensure that classified information is being encrypted correctly. If the device detects a problem with the encryption process it shuts itself down and illuminates a 'fault' LED. From this state the operator can restart the device by pressing the 'fault reset' button. Finally, so that the operator can test the device's self-diagnosis feature, a 'fault insert' switch is provided to deliberately cause the internal check to fail—activating the fault insert switch should cause the device to immediately shut down.

3.2 Initial Block Diagram

A block diagram of the cryptographic device is shown in Figure 3. Solid lines represent 'data' connections and dashed lines represent 'control' connections. Unidirectional connections are shown with arrows; other connections are assumed to be bidirectional.

The diagram is closely based on one in the device's technical documentation (Graves, 2003). (It omits the device's power circuitry and associated components.) The level of abstraction varies in the figure. Some of the components shown represent individual electronic components (e.g., component *I* is a single physical switch), whereas others represent entire circuits (e.g., component *G* comprises nine logic gates). Similarly, some of the connections represent individual pin-to-pin

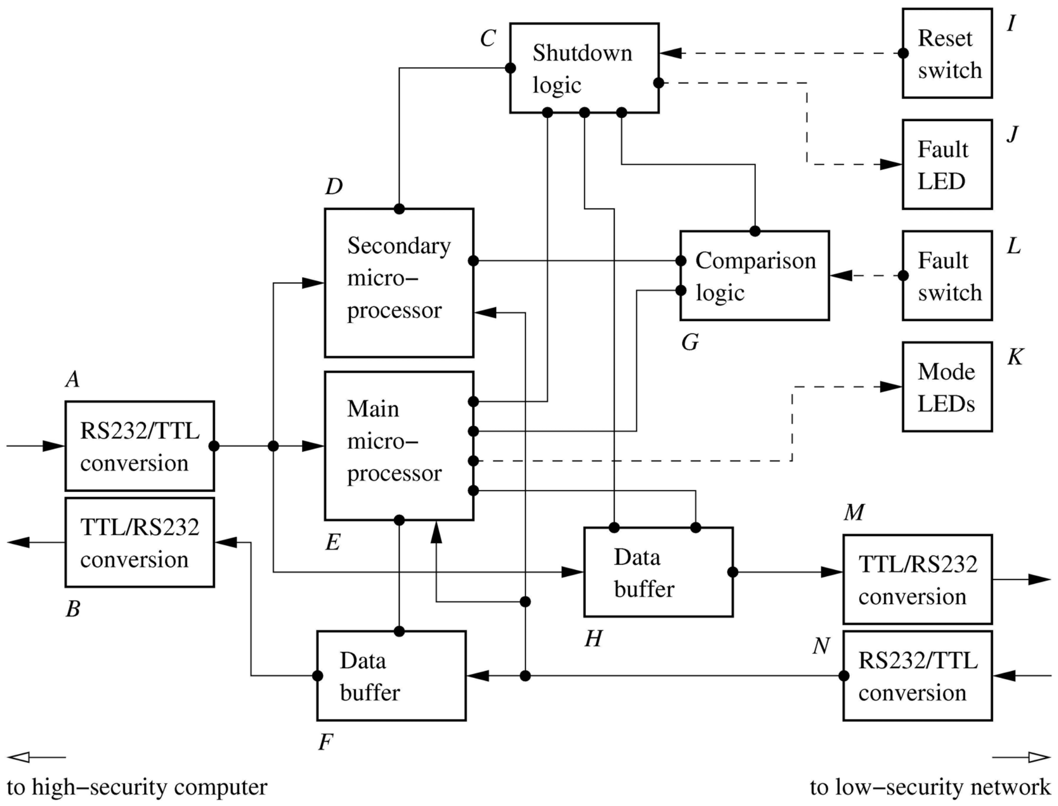

Figure 3: Initial Block Diagram of the Cryptographic Device

wires (e.g., from circuit *C* to LED *J*), while others are sets of parallel wires (e.g., the parallel connection from microprocessor *E* to circuit *G* carries an eight-bit byte from an output port on the microprocessor).

As a starting point, we have assumed here that the components on the device's periphery, and their associated connections, are already well understood. For instance, it is reasonable to expect that a connection to one of the LEDs on the front panel (e.g., component *J*) consists of an outgoing 'control' signal. Similarly, we expect that the RS232-to-TTL converters (e.g., component *A*) produce incoming 'data' values.

However, we assume that the device's internal behaviour has yet to be evaluated from a security perspective. Thus, all internal connections are shown as bidirectional and capable of carrying classified information, and all internal components are assumed to potentially contain classified information, until it can be proven otherwise.

3.3 Functionality

In this section we summarise the device's overall behaviour, as described in its accompanying documentation (Graves, 2003).

Components *A* and *B* in Figure 3 together form the device's connection to the high-security computer. They convert between the serial RS232 bit stream and the TTL voltages used within the device. Importantly, component *A* is assumed to be a source of red data from the high-security domain. Similarly, components *M* and *N* comprise the connection to the low-security network. In particular, component *M* is assumed to be a data sink leading to the low-security domain.

Components *I, J, K* and *L* are the switches and indicators on the cryptographic device's front panel. Their connections are all assumed to carry red control values, because the components manipulate simple signals only and because they all reside within the high-security domain.

The device's behaviour is largely under the control of microprocessor *E*. Under normal circumstances, classified information from incoming RS232 converter *A* is passed to microprocessor *E* for encryption. The resulting encrypted information is then sent into data buffer *H*, and from there to outgoing RS232 converter *M* to be forwarded to the low-security domain.

The classified information is encrypted by software within the microprocessor, using one of three algorithms, depending on the device's encryption mode. However, in the fourth 'bypass' mode classified information from component *A* is allowed directly through buffer *H* to output *M* without encryption.

In the opposite direction, unclassified information received by converter *N* is sent to microprocessor *E* for decryption by one of the three algorithms. The result is then forwarded to buffer *F*, and from there to the high-security domain via output *B*. Again, the bypass mode allows information to flow directly from *N* to *F* to *B*, without decryption.

The remaining components are all associated with the device's self-diagnosis capability. Redundant microprocessor *D* duplicates all encryption of classified information performed by microprocessor *E*. (Ideally this would be done using independently-developed software.) Both microprocessors send each encrypted character to 'comparison logic' circuit *G*, which produces a signal indicating whether or not the characters match.

This signal is interpreted by 'shutdown' circuit *C*, which, if a mismatch is detected, tells data buffer *H* not to forward information to the low-security domain. In addition, fault insertion switch *L* and reset switch *I* allow the operator to control the self-diagnosis features as described above. (For clarity we have separated switch *L* from logic component *G*, although the device's detailed schematic diagram (Graves, 2003) reveals that the switch is actually embedded within the logic circuitry.) Finally, the *D–C* and *E–C* connections are used to synchronise the microprocessors' outputs with the shutdown circuitry.

Even at the block-diagram level, the device's design is complex. Ultimately, however, a high-grade information security evaluation must proceed down to the level of the device's schematic circuit diagram and microprocessor software. Therefore, the practical challenge is how to determine which parts of the device need to be studied in depth and which can be ignored or treated cursorily.

4. TECHNIQUES FOR IDENTIFYING CRITICAL COMPONENTS

Given a device model such as that in Figure 3, an information security evaluator needs to determine whether or not information from a 'red data' source can reach a data sink in the low-security domain. Doing so efficiently is aided by having ways of eliminating irrelevant components, and deciding which is likely to be the most efficient order for evaluating components.

4.1 Eliminating Irrelevant Components

The first thing the evaluator wants to do to simplify the evaluation task is to eliminate components that do not need to be examined at all. We therefore observe that those components that may have security significance can be defined as those that lie on an information flow path from a red data source to a data sink in the low-security domain. This is the largest set of components that the security evaluator will need to analyse, in the worst case.

In Figure 3, for instance, two possible sources of red data are incoming RS232 converter *A*, which is in the high-security domain, and microprocessor *E*, which acts as an upgrader when decrypting information. The only sink for information going to the low-security domain is outgoing RS232 converter *M*. (Switches *I* and *L* are sources of red *control* signals only, so we ignore them for now, but see the discussion on covert channels in Section 5.) Therefore, we are interested in all components on paths from components *A* and *E* to component *M*. Relevant paths include the following. (There is no need to list paths that iterate around the same loop, e.g., *D, C, G, D,* more than once.)

- *A, D, C, G, E, H, M*
- *A, D, C, H, M*
- *A, D, G, E, H, M*
- *A, E, H, M*
- *E, H, M*

The union of the components in these paths then gives us the largest set of potentially security-critical components: *A, C, D, E, G, H* and *M*. These components are shown in Figure 4, with all other components and their associated connections omitted. Figure 4 thus isolates the security-critical part of the block diagram in Figure 3.

This simple technique quickly reduces the number of components to be evaluated. In general, it eliminates the following types of component:

- Components whose outputs cannot reach the low-security domain. In our example buffer *F* and RS232 converter *B* may both potentially receive classified information from microprocessor *E*, but they can nevertheless be eliminated because their outputs return to the high-security domain. (Similarly, LEDs *J* and *K* can be eliminated both because they reside within the high-security domain and because we are assuming that their control-valued inputs are not security-critical.)
- Components whose inputs cannot originate from a red data source. In Figure 3 the only such component is RS232 converter *N*, whose sole input comes from the low-security domain.

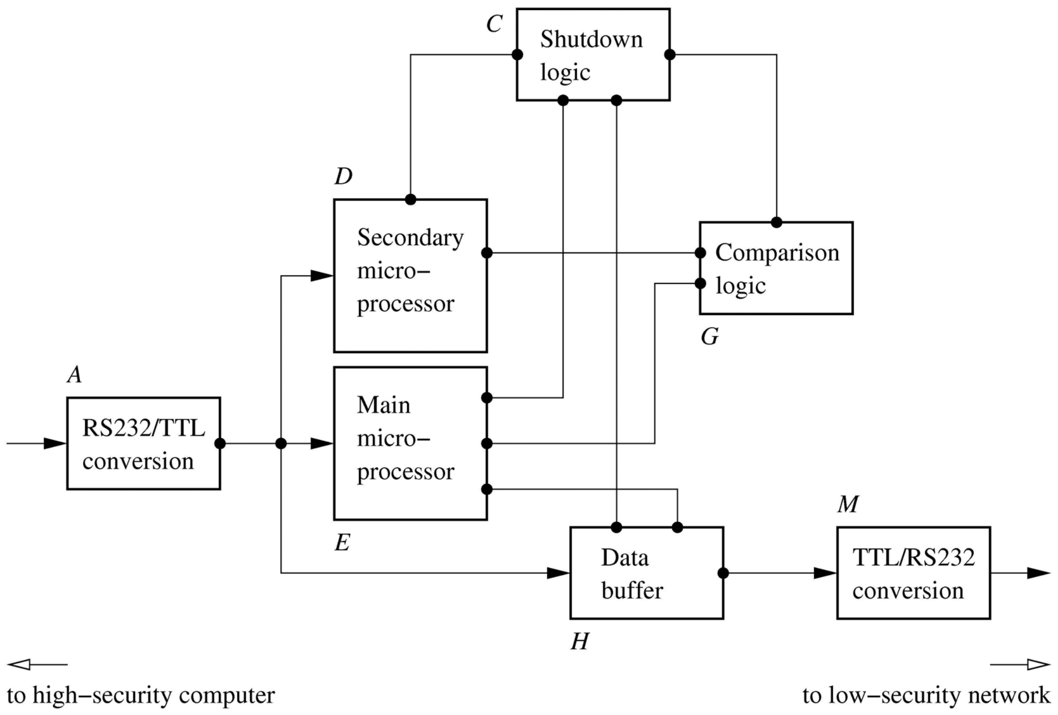

Figure 4: Potentially Security-Critical Components

4.2 Identifying the Most Critical Components

The evaluator's job can also be made easier by prioritising the components to be examined in depth. In particular, we observe that a simple and potentially highly-efficient approach is to adopt the graph's *minimal cutset* as the first group of components to be evaluated. Recall that the cutset of a graph is a set of nodes whose removal will make the graph disjoint (Provan and Shier, 1996). A minimal cutset is such a set with the smallest cardinality. (There may be more than one minimal cutset for a given graph.)

In our application we are interested in cutsets where one partition contains all red data sources and the other partition contains all data sinks in the low-security domain. This is an attractive approach because if it can be shown that the components in the minimal cutset achieve security domain separation, then the evaluation need proceed no further. In many cases, therefore, evaluating the minimal cutset first will result in evaluation of the smallest possible set of components.

In Figure 4, we want to find a cutset that separates red data sources *A* and *E* from low-security data sink *M*. In this case there is a unique, singleton minimal cutset consisting of data buffer *H*. In other words, if we can show that this single component blocks transmission of red data then the entire security evaluation is complete.

4.3 Evaluating a Component

The next step is to evaluate the security-critical components in the chosen cutset. The goal is to show that these components have the necessary security functionality, i.e., that they are all either downgraders or data sinks. If so, then the evaluation can stop because the device is guaranteed not

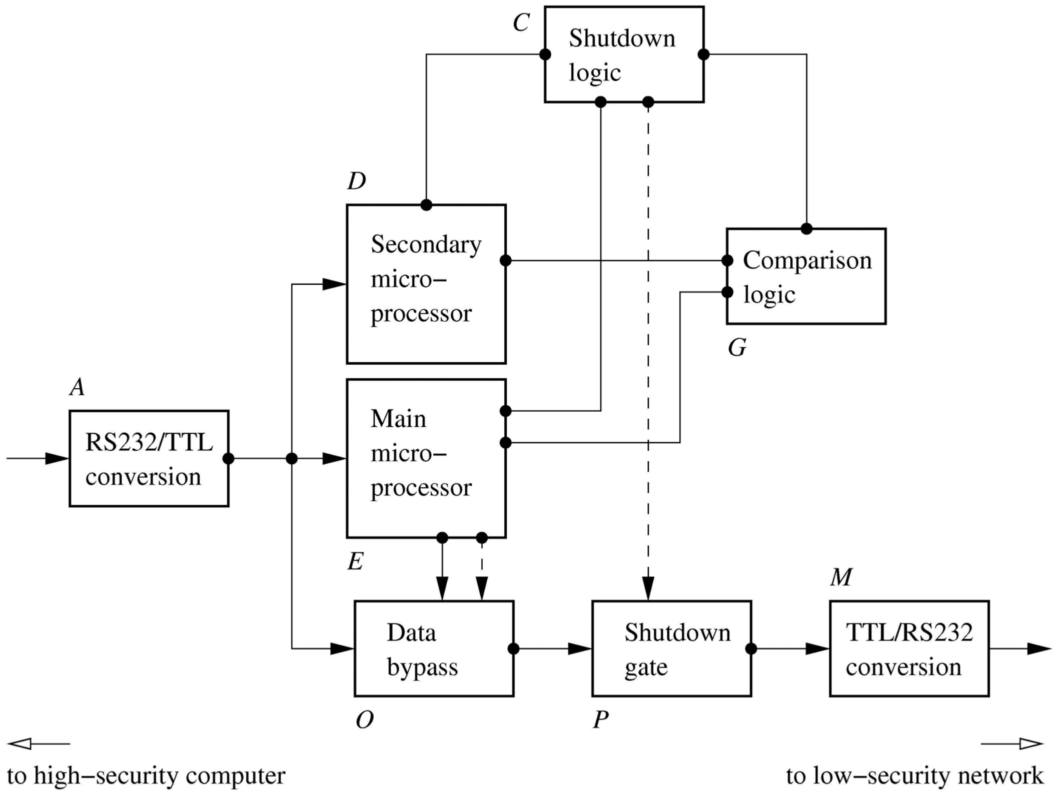

Figure 5: Reevaluated Block Diagram

to send red data to the low-security domain. Otherwise the results of the component evaluations are inserted back into the diagram and the whole process repeats.

Above we identified buffer *H* as the first component worthy of close examination. A study of the cryptographic device's documentation for the buffer (Graves, 2003) reveals that it is composed of two parts. The first, a 'data bypass' circuit *O*, forwards characters received either from microprocessor *E* or directly from RS232 converter *A*. It also responds to a signal from microprocessor *E* which controls the release of characters, depending on the encryption mode. The second component, a 'shutdown gate' *P*, can block the transmission of characters to the low-security domain, as directed by shutdown circuit *C*.

Figure 5 shows the block diagram with newly-evaluated components *O* and *P* inserted in place of component *H*. The previously assumed bidirectional data connection between *E* and *H* is now known to consist of two separate unidirectional connections to component *O*, one carrying (potentially red) data and the other carrying a control signal. Similarly, the connection between components *C* and *H* is now known to be a control signal sent by component *C*.

Continuing the evaluation, we note that the minimal cutset of Figure 5 consists of component *P* alone. The device's schematic diagram (Graves, 2003) reveals that component *P* is a single tristate buffer gate with an inverted enable input. It may thus allow data to pass straight through, depending on the disabling signal from shutdown circuit *C*. (Notably, since this control signal is an inhibitory one, a failure in circuit *C* would allow component *P* to forward potentially classified information to the low-security domain!)

Therefore, component *P* alone is insufficient to guarantee security. We can thus remove it from the graph and connect component *O* directly to component *M*. (The *C–P* control signal is assumed here not to be capable of carrying red data and can also be eliminated.)

Reevaluating the resulting graph then gives us a minimal cutset consisting of components *E* and *O*. The device's detailed schematic shows that data bypass circuit *O* consists of three logic gates and acts as a switch (with two data inputs) under the control of microprocessor *E*. Again, therefore, this circuit alone is insufficient to guarantee security.

Thus, our final conclusion is that the ultimate responsibility for the flow of information from red data source *A* to data sink *M* rests with the software running on microprocessor *E*. This program will therefore need to be subjected to careful scrutiny to ensure that it enables the release of classified information appropriately. (Relevant information-flow analysis techniques for security software exist (Avvenuti *et al*, 2003), but are outside the scope of this paper.)

Pleasingly, this final outcome matches our intuitions about the device's design. In its three main encryption modes, microprocessor *E* is meant to act as a downgrader, effectively stopping the flow of red data, and thereby establishing the necessary security functionality. In the fourth 'bypass' mode, microprocessor *E* again has responsibility for correctly authorising component *O* to release (unencrypted) red data.

Most importantly, with respect to Figure 3, the overall evaluation process did not require us to examine microprocessor *D*, circuits *F*, *C* and *G*, or components *B, N, I, J, K* and *L* in any depth, thus saving the security evaluator a considerable amount of work.

4.4 A Generic Security Evaluation Process

Abstracting from the case study above allows us to define a straightforward process for efficiently structuring high-grade information security evaluations.

1. Begin with a block or circuit diagram of the device and identify all red data sources (i.e., data sources in the high-security domain, including those within the device itself) and relevant data sinks (i.e., those leading to the low-security domain).
2. Apply the path-traversal technique from Section 4.1 to eliminate components that do not connect red data sources to low-security data sinks.
3. Find a minimal cutset, as described in Section 4.2, which partitions the data sources and sinks, and thus identifies the component(s) to be examined first.
4. Examine each of the components in the chosen cutset:
   (a) If all the components in the cutset can be shown to block propagation of red data, then the device is secure and the evaluation is complete.
   (b) Otherwise, substitute the results of the component evaluations back into the diagram, and return to step 2.

5. DISCUSSION

For the purposes of illustration, we assumed above that 'control' information is not of security relevance, and we thus did not consider 'red control' signals in the security argument. This is often a reasonable assumption in practice. For instance, 'comparison logic' circuit *G* in the cryptographic device actually has two 'data' inputs, from microprocessors *D* and *E*, and produces a 'control' output, to ‘shutdown logic’ circuit *C* (Graves, 2003). Each of the inputs consists of an (encrypted) eight-bit character, whereas the output consists of a single binary signal only. Therefore, even if the microprocessors failed to encrypt their outputs, and the inputs to circuit *G* thus carried classified information, there is still comparatively little information flow through the circuit.

Nevertheless, the assumption that control signals are of no interest is not necessarily true in the field of *covert channel* analysis (Bishop, 2003). One of the goals of covert channel analyses is to determine whether or not information can be conveyed in ways unintended by the device’s designer, including via ‘control’ signals. Fortunately, the techniques we described above can be readily adapted to covert channel analyses, simply by treating ‘control’ connections to be as important as ‘data’ ones. For instance, in a covert channel scenario, switches *I* and *L* would both be treated as red information *sources*, modelling the (highly unlikely!) situation where classified information is leaked to the low-security domain via the pattern of operation of these switches. For example, someone in the low-security network could conceivably detect the interruption to the flow of characters caused by the operator flicking the ‘fault insert’ switch.

Another way in which this work could be extended would be to separate the analysis into different cases for different operating modes of the device. Certain components and connections could be included or excluded depending on whether or not they are expected to be active in certain modes. For devices with a number of distinct modes, this would make the overall evaluation clearer and easier because the evaluator would need to consider information flow in only one situation at a time.

6. CONCLUSION

High-grade information security evaluations are difficult and expensive. We have shown how the process can be made more efficient by applying simple, well-known graph analysis techniques to both eliminate components that do not need to be evaluated at all, and to choose those components whose evaluation is most likely to lead to the quickest outcome. In the case study, over half of the device’s components were eliminated from consideration, without the need to gain a detailed understanding of their function at all.

Importantly, the techniques for eliminating and prioritising components in Sections 4.1 and 4.2 are readily automatable. Indeed, we have already developed a prototype tool that allows device diagrams to be entered graphically and then applies the techniques described above to highlight those components worthy of evaluation.

ACKNOWLEDGEMENTS

We wish to thank Andrew Matthews and Scott Mallen for their technical advice on ‘critical regions’ in information security evaluations. We also wish to thank Luke Wildman for his many helpful technical suggestions during the course of this work, and Tim McComb for developing the prototype graphical analysis tool. This research was funded by the Defence Signals Directorate and the Australian Research Council via Linkage-Projects Grant LP0347620, *Formally-Based Security Evaluation Procedures*.

REFERENCES

AVVENUTI, M., BERNARDESCHI, C. and DE FRANCESCO, N. (2003): Java bytecode verification for secure information flow, *ACM SIGPLAN Notices* 38(12): 20–27.

BISHOP, M. (2003): *Computer Security: Art and Science*, Addison-Wesley.

COMMISSION OF THE EUROPEAN COMMUNITIES (1991): *Information Technology Security Evaluation Criteria (ITSEC)*, 1.2 edn.

COMMISSION OF THE EUROPEAN COMMUNITIES (1993): *Information Technology Security Evaluation Manual (ITSEM)*, 1.0 edn.
COMMITTEE ON NATIONAL SECURITY SYSTEMS (2003): National Information Assurance (IA) Glossary, Instruction No. 4009.
GRAVES, J. (2003): Cryptographic device, Technical report, Defence Signals Directorate.
PROVAN, J. and SHIER, D. (1996): A paradigm for listing (s,t)-cuts in graphs, *Algorithmica* 15, 351–372.
THE COMMON CRITERIA PROJECT SPONSORING ORGANISATIONS (1999*a*): *Common Criteria for Information Technology Security Evaluation*, 2.1 edn. ISO/IEC Standard 15408.
THE COMMON CRITERIA PROJECT SPONSORING ORGANISATIONS (1999*b*): *Common Methodology for Information Technology Security Evaluation*, 1.0 edn.
YOUNG, W. D. (1991): Verifiable computer security and hardware: Issues, Technical Report 70, Computational Logic Inc.

BIOGRAPHICAL NOTES

Andrew Rae is a systems assurance engineer for Invensys Rail Systems Australia. He has previously worked for the University of Queensland, Massachussetts Institute of Technology, and the Australian Department of Defence. His interests include safety and security evaluations for software-intensive systems, automated fault tree analysis, and preparation of safety cases.

Andrew Rae

Colin Fidge is Professor of Computer Science in the School of Software Engineering and Data Communications, Queensland University of Technology. He completed his PhD at the Australian National University. His research interests include security evaluations, maintenance of legacy programs, and high-integrity software engineering.

Colin Fidge