SimpSA

A Simple Security Architecture Methodology

By

Tony Bull

Notitia Salu Ltd.

This book is dedicated to:

All those lonely security engineers, security analysts and security architects, who slave away trying to perform an objective security analysis, knowing full well, it is always subjective.

Table of Contents

Introduction 1
What is a Security Architecture? 1
Why have a Security Architecture? 1
What does a Security Architecture need to include? 2
The SIMPle Security Architecture 2
SimpSA: Challenges in Building a Security Architecture 3
Complex Security Architecture Methodologies 3
The Documentation Challenge 3
Chapter One: The Link with Risk Management 5
Chapter Two: SimpSA and Projects 10
Chapter Three: A Sample SimpSA and Projects 13
Epilogue/Conclusion 15
Bibliography 16
Acknowledgments 16
About the Author 17

Introduction

What is a Security Architecture?

A Security Architecture is a high-level design document laying out how the major security risks to an information system are mitigated.

This definition of a Security Architecture is generally accepted across the Information Technology sector, though it is not defined in any formal sense and is therefore often subject to misinterpretation. However, in this document, this definition of the Security Architecture objective is a firm and fixed definition that underpins the SIMPSA methodology.

Why have a Security Architecture?

People are familiar with the original use of the term "architecture", as it is applied to buildings. The "architecture" was the broad outline of a building design. A top-level statement of how the many detailed aspects of the building will building work together, to create a functional, coherent building to meet its users' needs.

"A major function of architecture as a tool is to Manage complexity in large projects".

"Enterprise Security Architecture, A business Driven Approach"

Sherwood, Clark, Lynas

The aspect of Architectural design that is useful here is the capturing of a top-level design vision, that is not bogged down in too much detail, but that allows many smaller distinct projects and activities to produce a complex but still coherent end product. As such the "Architecture" paradigm has been adopted in many industries where complex programs, require many sub-programs and projects to run in parallel, to produce the final end result.

Architecture in IT has developed to the point that it is not represented as one single view of the system. Because of the abstract nature of complex computer systems, it has become the norm to represent the Architecture in a series of views of the system.

A good example of architectural views are those codified in "The Open Group Architecture Framework" (TOGAF) as:

- Business Architecture
- Data Architecture
- Applications Architecture

- Technical Architecture

Other commonly developed architectural views, that help systems development include:

- Physical Architecture (based on physical topology);
- Logical Architecture (based on logical topology);
- Network Architecture (combination of Physical and Logical for connectivity);
- Services architecture (process mapping); not forgetting
- Security Architecture

The last entry, Security Architecture, can now be seen as a complementary view of the system(s), that provides a top-level view of how security is supported in the system and is used to provide the top level vision of how the system will be kept secure.

What does a Security Architecture need to include?

It is widely accepted that "nothing is free from all risk", so nothing is completely secure. In this context it is also widely accepted that the hunt for perfect security is ruinously expensive, so the most appropriate security can only be achieved by good risk management.

Risk management assesses risk to valuable assets and decides whether to accept, avoid, share or reduce the risk. The latter often offers the most sensible approach, as it is often cost effective in IT to reduce a risk by deploying controls. The logical conclusion is that a Security Architecture that identifies the main controls and links them to the risks, would be a helpful document in showing how systems are optimally secured.

The SIMPle Security Architecture

This document describes a simple methodology for creating a Security Architecture that is simple to generate and accessible to non-security professionals. It does this by simplifying the theoretical risk management approach, to remove intermediate steps that in practice are not always especially useful.

SimpSA: Challenges in Building a Security Architecture

Complex Security Architecture Methodologies

The theory of a Security Architecture can quickly become quite complex and arcane. There are many the Author has had the chance to use [HMG IS1 (UK Government), DBSy (UK), EBIOS (Fr), Threat Scenario Modelling (EU), SABSA (USA)] and all required dedicated security professionals, usually specifically trained in the methodology, to stand a chance of successfully using the process.

SABSA is a good example. The textbook of the methodology is a dense 600 page hardback book, backed up by 6 different multi day training courses costing many thousands of £pounds.

All the techniques I have used are viable and useful, if they are properly resourced. However, it is the Authors experience that they rarely are resourced adequately. Worse, because they can delay the front end of a project and produce masses of dense paperwork, they often suffer from failing management commitment to maintain the resources throughout the project. This means in the latter stages of projects, the process is given lip service only, with the compliant analysts being forced to manipulate the results to provide the required outcome.

If this is not a fair characterization of your organisation, you may well be better served choosing a more established and rigorous approach, such as SABSA. This is especially the case if your systems have some high value assets needing high assurance of security.

If your organisation is in a rush, is deploying "agile" development programs, if you will struggle to obtain and retain skilled security architects, the SIMPSA approach should help you to keep your project suitably secure, without those specialists security architect resources in abundance.

The Documentation Challenge

For as long as I have been in engineering, there has been a recurring "cri de coeur" to document systems better. Over that same period, the world has changed immeasurably by the development of the Internet and all the amazing on-line systems that have moved so much of out lives online, often with very little in the way of supporting documentation.

The large majority of projects do a small amount of documenting at the front of the project, then fail to maintain the documents, which become out of date. Even on projects that have devoted substantial amounts of effort to documenting the systems, there is always a general paucity of up to date and accurate information.

There is good reason for this problem. The vast majority of documentation is still provided in the form of the written word (often the lingua franca of technology, English).

The problem is that English is not a good medium to accurately define complex technical products, especially when articulated by engineers, who are often poorly disposed to good communication, and even worse so when that engineer is not using his or her first language.

As such it is important that the documentation that is required is kept to a minimum and as simple as possible, which is one of the strengths of the SimpSA approach.

Chapter One: The Link with Risk Management

Tackling the problem backwards

If the main objective of a security architecture is to link risks and security controls, it may seem sensible to start the security architecture analysis with a given set of risks. The problem with such an approach is that it can lead to the infamous predicament, so wonderfully summed up by the Irishman when asked directions to some local landmark who said "I wouldn't start from here".

Identifying the risks against systems of only moderate complexity can quickly become a monumental task. The "completeness" imperative requires that all significant risks be found, as it will be a very poor analysis that misses a risk that is later realised.

This challenge means that the analyst has little option but to accept all risk postulated by stakeholders. The project director might postulate that a firework could land on the datacentre and cause a fire. The Lead architect might insist that "buffer overflow" is a concern. The Security Engineer is worried about slow patching. The accreditor worries about the OWASP top ten vulnerabilities. The CEO is worried about the latest ransomware attack.

When analysed, it becomes clear that each risk requires lateral analysis. For example, the "firework", could land in lots of places, which could cause all sorts of different problems if they occur. Another example would be the OWASP top 10. This would be the latest top 10, which could create 10 risks. But a more rigorous approach would be to cover all the OWASP top 10 from the past 10 years. Also, the OWASP top 10 relates to the external Web interface. Internal trusted interfaces have more powerful APIs, so are more vulnerable if there is an insider attack, requiring another replication of the risks relating to different interfaces.

Definition of Risk

There is no generally accepted definition of the term risk. Some theoretical methodologies like to start from the perspective of an attacker with motivation to compromise an asset. This creates a risk in principle that needs to be mitigated. The problem with this in modern systems is that accidental compromise and collateral damage is as big a problem as a motivated targeted attack. Similarly, most lay folk would agree that a risk, that is fully mitigated is no longer a risk.

In this document, RISK is defined as occurring where credible threats to assets (accidental or malicious), are not mitigated fully, or put another way, there are vulnerabilities in the systems defences!

Hopefully this gives the picture of how a risk analyst may feel that all risks are afflicted by the "Gemino curse" from Harry Potter, where everything he touched in Gringotts bank vault kept multiplying. The author has worked on several significant systems and found that a "tight" analysis of systems can lead to 500+ risks. One system worked on for over 10 years, was analysed solidly by a team of 10+ over that period, and they tried to manage in excess of 7000 risks. Most of the time it felt like a job creation

activity. I never felt it brought any trustworthy conclusion to the question "have we found all the risks".

Part of the problem is that the ontology of risk is often not rigorously applied. In the example above, the risks have different ontologies that clash:

• A buffer overflow is a detailed technical vulnerability, of the type captured in the OWASP top 10. Having it and the OWASP top ten in the risk list suggests an incomplete approach to identifying vulnerabilities.

• A ransomware attack is an attack. An attack will fail if there are no vulnerabilities. So are attacks risks or are vulnerabilities risks?

In conclusion, it is the authors experience that the majority of risk assessment techniques are fraught with sprawling over complexity and bogus rigor. They often add little value to projects, especially if rushed or poorly resourced in other ways.

This conclusion brings into question the merit of security analysis, because if you do not have a reliable risk list, it is not possible to manage the risks effectively. But that is the wrong conclusion: We should reverse the problem and develop a representative and manageable risk list from the controls, instead of the other way round. Essentially this is tackling the problem backwards and it leads to a simple and effective approach.

Working back from Control Lists

There are many examples of frameworks, governance regimes and guidance documents, replete with a wealth of knowledge about what security controls should be used to make systems (e.g. ISO 27001, CCM, NIST, CE+ etc). For most systems, it is a reasonable assumption that faithful implementation of all applicable controls, will lead to an acceptable level of residual risk to the intended system.

This basic assumption leads to a powerful way to identify risks. The Security Architect must analyse all the controls in appropriate governance frameworks and identify all that are applicable. Any that are judged to be applicable, but are not adequately implemented, are captured as a risk. In this case, the risk assessment framework is tightly bound to implementation of the control, achieving the goal of a Security Architecture to link these 2 attributes: risk and security control.

This approach means that the first job of the Security Architect is to determine whether a control is applicable or not, within the context of the specific system to be protected.

As the design progresses, the Security architect also must consider assurance. Assurance in this case is the confidence in a control to dependably enforce its function. In this architectural approach, assurance is a simple overlay. For example, a Security Architect

may consider that satisfying a boundary protection requirement using a single firewall is inadequate, because the assets protected are too valuable to trust to the competence of one manufacturer. This implies a 2nd firewall is required, deployed in addition to the first, or a full DMZ. If there is no 2nd firewall, the implied vulnerability is captured as a risk.

The method can be extended in terms of justifying the rigor of finding all risks and ability of an organisation to show due care when protecting their systems. The collective wisdom contained in existing standards and guidance on advisable controls is considerable. By choosing to spend more time analysing controls in documents other than the regulatory mandated standards, the organisation can show due care was exercised. There is also the possibility to go beyond regulations, standards and guidelines, by considering the latest industry developments in controls.

For example, at time of writing, Windows 7 is no longer available and Microsoft are rolling out Advanced Threat Protection across most users of Windows 10. ATP does not appear in most standards. A Security Architect can add a risk to a system still running windows 7, that it does not have ATP and therefore subject to greater risk of APT attack. This may be mitigated by existing controls such as SIEM, IDS, IDP or DLP tools, but it may not. Analysis of ATP capability will tell the Security Architect if security inflation since the deployment of Windows 7 has, in effect, created a new risk for the enterprise.

Moving from a list of Controls to a Security Architecture

At the heart of the SIMPSA approach is the simple but powerful assertion that securing systems is about deployment of cost effective security controls. This is embedded in the approach by dictating that each and every risk is expressed as a measure of the effectiveness of one or more controls.

For example:

Control	Risk
Access Control by Single factor complex password	There is a MEDIUM risk that users will share passwords and loose them in a phishing attack.

The advantage of this approach is that it provides a direct link between a risk and the control that mitigates the risk. This in turn allows the cost of the mitigation to be weighed against the risk, so that proportionate and informed decisions on risk mitigation can be made.

Linking Risk and controls is not constrained to being a 1-2-1 linkage. Sometimes, a risk will need many controls to be fully mitigated. In the above example, the risk identified may be further mitigated by the deployment of a policy precluding the re-use of passwords on different systems and a technical control, randomly generating passwords for users.

Similarly, controls may mitigate many risks. In the above example, a complex password for access control will also mitigate a range of password attacks from insiders and outsiders. So in conclusion, it may be necessary to implement a "many to many" relationship between risks and controls. This can be done in by the use of a simple database, with a table for risks and a table for controls, then cross referencing the 2 tables.

However, this methodology is best started simply, using a single spreadsheet, by limiting risks having a single list of controls and link to 1 or 2 risks at most. Additional complexity in the "many to many" linkages can be added later as additional rigor is required.

The place of Threat and Asset in SimpSA

HMG Infosec Standard Number One, elevated the theory of risk management to the point that it did not even consider vulnerability, nor any aspects of the security controls in order to generate a risk list. In the IS1 approach, a valuable asset implies risk. An asset is threatened by a Threat Source, which motivates a threat agent to attack using a range of codified methods.

This approach had it's advantages, but was not effective in practice, and has now been deprecated because of the lack of utility in the UK government sector. It is one of the methods that often led to bogus rigor. Lots of work compiling reams of documentation that didn't actually help much in securing the system.

For this reason, the SIMPSA approach does not explicitly include Threat in the analysis. This is one of the simplifications that makes SIMPSA simple. However Threat is not ignored. Security controls have only been developed on the basis of Threats successfully exploiting vulnerabilities. As such Threat can be considered to be included implicitly, as embodied in the security controls.

Similarly, Assets are not explicitly considered in the SIMPSA approach. This again helps with keeping the approach simple. The analysis assumes a "system high" approach. Risks are graded against the most valuable assets in the systems being designed. Controls are enhanced for very valuable assets, or the risk is rated higher, where the control is inappropriately weak for protecting a valuable asset.

The lack of explicit use of Threat and Asset in the SIMPSA approach is necessary to keep the approach simple. It means that the approach is perhaps not rigorous and complete enough for all systems. However, as stated elsewhere in this document, it is wise to be aware that rigour can often be difficult to achieve in practice.

Estimating Risk Size

In the above example, a risk is estimated as “Medium”. The way this value is estimated is largely out of scope of the SimpSA methodology. If required, the analysts can

bring in the usual primitives that constitute a mathematical risk estimate: asset value, threat assessment, vulnerability and likelihood. However, the Authors experience is that these methods often flatter to deceive, with their pseudo-scientific nature misleading stakeholders into believing the output is objective truth rather than a subjective view. It can be better to embrace the subjectivity, to enable a debate in which those with most knowledge of the system can contribute.

Chapter Two: SimpSA and Projects

Working SIMPSA into Projects

Most organizations accept that security should be considered from the start of a project, but the truth is that a lot of security experts get a bit lost in the fuzzy front end of projects. Security is not a "thing". It is an attribute or characteristic of a thing. In the project "fuzzy front end", it can often be difficult to make progress with security analysis, because of the lack of definition of what the thing, or widget the project is trying to create. Getting security involved too early can cause project constipation, as they start adding red lines to pages that lock down design possibilities.

In the early stages of the project, it is important to be able to pick out the major costly controls and start to blend them into the design of the system. It is also important to retain flexibility to support the growth of definition of what the “thing” is it is supposed be.

The SimpSA approach supports this by leading the Security engineer, architect, or designer to obtain a list of controls from standards and guidance. This supports an initial "quick and dirty" analysis to provide a framework in the early stages of the project.

As the project progresses, additional resources and effort can be deployed and additional diligence is required to ensure the completeness of the controls and assure the final system’s security. At this stage it is important to have an extensible security architecture, capable of absorbing additional risks and supporting greater rigor in the analysis of risk mitigation.

This flexible approach is important in lots of projects, but also with Agile developments, whereby products evolve organically, requiring any security analysis to be rapid, interactive and accessible to the non-security professional.

The accessibility of the analysis is another key attribute of the SimpSA approach. The simple trail (in a spreadsheet), between a control and the risks it is mitigating, allows for greater transparency in the design process, and better designs. This avoids the "security huddle" where security SMEs use their own arcane and impenetrable notations that excludes other specialisms, and which can appear arbitrary or like “black magic” outside the security bubble.

The methodology for doing this can be tailored to skills and procedures available to the enterprise. Tools such as Domain Based Security modelling System (DBSY https://en.wikipedia.org/wiki/Domain_Based_Security), though it is not necessary to follow a formalised approach such as these. Tools that can be interpreted by the design authority for the system should be chosen. The important aspect of the architecture is that it continues to show the link between controls (e.g. firewalls, personnel vetting, door locks) and the risks that occur when they are inadequate.

The SimpSA advantage for Agile, DevOPS and CI/CD development

In DEVOPS, CI/CD and Agile system development, the focus is on fast deployment of useful additional bits of a product. This is typically done in Sprints, which are fully contained development projects (requirement, design, develop test, deploy) with small useful additional product features such as microservices delivered. For Agile, small development teams are formed into a "Scrum", with much autonomy on how they get the sprint done in the typical 2 week cycle.

Providing security guidance to these teams is difficult. Often it is not possible to assign a single specialist security subject matter expert to all the Scrums. In this situation it is important to have a security governance regime that supports the Scrum process by:

1. Providing an accessible justification for why a security control is needed, so that all scrum participants can understand it.

2. Can be iterated fast, to take account of changes from unexpected outputs form Scrums.

3. Is powerful enough to provide meaningful feedback on security vulnerabilities fast, and justify scrums changing "mid-sprint".

SimpSAs simple approach supports understanding by the non Security professional, as well as fast iteration within the sprint timeline. The areas that SimpSA lacks, is rigor of analysis and being able to provide assurance that all significant risks have been considered. There are more rigorous approaches available, but what the security analyst needs to consider is whether the rigor is achievable within the constraints of a very dynamic Agile development project.

Simple Visual Representations for Management Reporting

SIMPSA is a based on the “Cause and Effect” risk assessment methodology. This is defined in ISO/IEC 31010 as one of the recommended risk assessment techniques to be used in information security risk management.

This methodology lends itself to representation the Ishikawa “fishbone” diagrammatic technique. In this technique, the bones of the fish skeleton branch out into a structured representation of the chosen security control framework. Wherever the control is insufficient, the control can be colour coded (Red Amber or Green) to show level of risk.

In the Ishikawa representation shown in Figure 1, the main bones of the “fish” are categories in ISO / IEC 27001 standard.

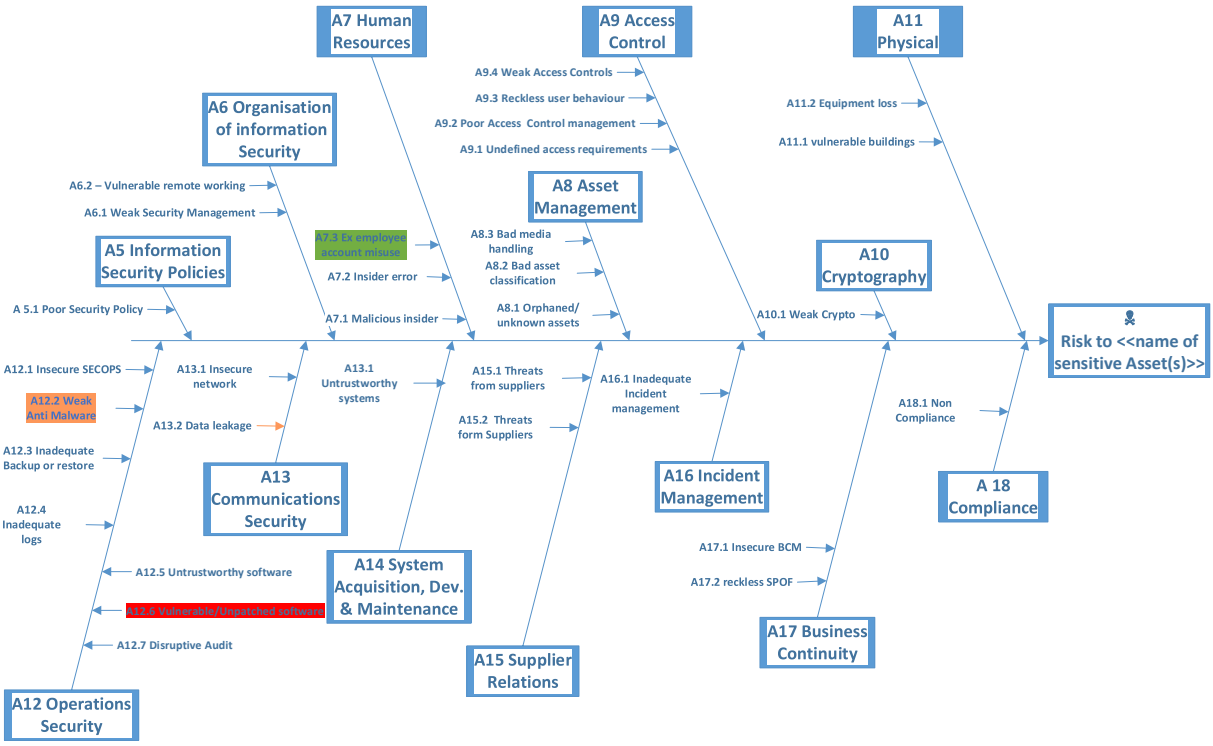

Figure 1 - Ishikawa Example Risk Map

In Figure 1, three notional risks are RAG labelled. A12.6 represent a High risk related to unpatched systems. A12.2 represents a medium risk due to incomplete anti-malware controls. Finally, there is a low risk account misuse by ex employees.

This approach is particularly useful when comparing the risks between different programs, when it can provide a quick visual indicator of the state of a project.

Chapter Three: A Sample SimpSA and Projects

The baseline implementation

The following control is a typical control:

SECREQ1 The organisation has a suitable perimeter security device.

The Security Architecture must identify which functions provide this perimeter security.

Control	Supporting Architectural Security Function
SECREQ1 The organisation has a suitable perimeter security device.	Internet gateway

A risk would be raised if an IT Health Check (ITHC) identified a problem that the protection was insufficient. For example: SECRSK1 the perimeter firewall is a packet filtering firewall is vulnerable to data loss because it does not inspect traffic.

Table 1 illustrates an example of how the security architecture provide the bridge between a control, a risk, the security function that supports that control and therefore the cost of risk mitigation.

Security Architecture		**Risk Management Activity**		
Control	**Control supporting Security Functions?**	**Risk Assessment**	**Remediation planning**	**Risk Management Decision**
.....	….	….	….	….
SECREQ1 The organisation has a suitable perimeter security device.	Firewall A is a packet filtering firewall that protects the Trusts connection to the Internet. Firewall B is an application-level gateway that protects the	SECRSK1 The perimeter firewall is a packet filtering firewall is vulnerable to data loss. (Medium)	Firewall A needs to be upgraded to an application-level firewall at a cost of £20k.	Balance the cost of £20k against removal of a medium grade risk.

	Trusts connection to HSCN.			
SECREQ2	….	….	….	….

Table 1 – Sample Analysis of a Security Control

In its baseline form, the Security Architecture can be as simple as multiple entries in the list shown here. The left-hand column is populated with all applicable controls. The list of potentially applicable controls can be obtained from standards (e.g. ISO27002). To keep the analysis simple, it is sensible to just capture the biggest risk for each control, but it is not too difficult to allow multiple Risks per control. If you wish to capture multiple Controls per risk, the analysis needs to be moved to a database tool to allow cross linking many to many from separate tables of Risk and Controls.

Enhancements to the Simple Security Architecture

The main advantages of this approach is the simplicity of the analysis and the accessibility to non security stakeholderss.

In its simplest form, the Architecture is a simple excel list of control, vs the prime technical component(s) used to provide the control. However, this can be a bit limiting, because often many security functions, across people, process and technology are required to fully satisfy the requirements of a mandated control.

Therefore, it is recommended that, as the security architecture maturity grows, it is expanded by deeper analysis on the full set of security functions deployed by the Trust to satisfy a control. This may require transferring to the Security architecture from tools such as Excel, to MS word or, in advanced cases, database or Computer-Aided Software Engineering (CASE) tools to capture the multi-table linkages that appear.

Epilogue/Conclusion

The Simple Security Architecture (SimpSA) is a simple way to link security controls back to the risks that they are mitigating. It can be captured in a flat spreadsheet and quickly allows the security architect to develop a compelling picture of the level of mitigation provided by the controls.

In order to use the methodology, the security architect constrains all risks to be expressed as a vulnerability in a system security control. Embedding this constraint in the approach, has 2 advantages:

• It makes the link between risk and the mitigating control, and the cost of the mitigating control very clear.

• It supports an effective hunt for risks, by analysing list of existing controls and deciding if the system implements them adequately, without vulnerability, or not.

- If not, a risk is implied.

The simplification in the approach have been shown to be effective, especially in immature, time and resource constrained organisations. The methodology is extensible and can be extended when time and resource allow, or more importantly, when the value of the methodology is established.

Bibliography

RISK

SABSA

Security Architecture

SimpSA

Threat/ThreatACtor/Threat SOurce

Acknowledgments

I would like to acknowledge SABSA, as developed by the Sherwood Institute as a truly valuable, innovative and thorough tool. It is well thought through and executed, but it took being assigned to assist a hopelessly immature and under-resourced organisation to realise the impossibility of getting SABSA implemented in many organisations. It therefore motivated me to formulate this simpler approach.

~~Thank the key people who inspired you and helped you throughout the process of writing and publishing your work. This is somewhat similar to the dedication page, except here you can elaborate and include more people.~~

About the Author

Tony Bull, MIET, C.E.Eng, CISSP, ISSAP, CCSP, CCP (Arch), is an security engineer, architect and subject matter expert with 30+ years' experience developing complex systems. This started in a radio R&D lab, developing military grade radio and transitioned via cellular development (infrastructure and terminals) into an abiding interest in computer security, information security, information assurance, Cyber security and "Cyber", where he has spent the last 20 years acting as the security lead and subject matter expert on a range of projects from high assurance national security products, to Aerospace CNI systems and commercial Web portals.

Tony's search across these last 20 years has been for the grand unifying theory of security, a theory that unfortunately has probably already be found: "security is about good risk management". Like many, Tony finds this an unsatisfactory answer, and hence Tony is still searching, but in the meantime has written this book to help those applying risk management to complex computerised systems. The intent is to capture the essence of 20 years experience analysing systems and applying complex technical controls in the least irrational and inconsistent manner possible and reach acceptance, that it will always be subjective.

