Open Reference Architecture for Security and Privacy

Asim Jahan

Maikel Mardjan

Open Reference Architecture for Security and Privacy

Version : 1.0

Date : 3-11-2015

Status : First Public Release

Table of Contents

About the authors

This first release of the open reference architecture for security and privacy is created by the following IT security architects:

Asim Jahan

Asim is passionate about helping companies to secure their business better as an independent Senior Information Security Consultant: www.jahan-is.com. He speaks, blogs and observes developments in cyber realms. Holding a Bachelor degree in Business IT & Management of The Hague University and knowing project management very well he likes to keep things practical using the Keep It Simple and Solid (KISS) method. Asim is married and adores his two beautiful kids. If he's not working he's playing with them. Or he's reading or following a course. And somehow he finds time for helping young talented bright students advance in their professional career: www.mythonline.nl. Take a look on his LinkedIn profile: https://nl.linkedin.com/in/asimjahan.He loves Metallica, just like his 4 year old son.

Maikel Mardjan

Maikel is a IT security architect and loves to make designs for complex IT systems in a simple way. Maikel holds both a Master (Msc) Business Studies of University of Groningen and a Master degree (Msc) Electrical Engineering, of Delft University of Technology. Maikel is TOGAF 9 Certified and CISSP (Certified Information Systems Security Professional) certified. Maikel currently works for the innovative IT company nocomplexity.com. Despite privacy concerns, Maikel can be found on Twitter too https://twitter.com/maikelmardjan

Foreword

Freedom is, was and will always remain important. This applies to our physical world as well as our digital world. To maintain our freedom we need protection and good IT security. Good security brings freedom to run your business the way you want, exchange information when needed and to keep secrets when needed.

Good security and privacy do not have to be endlessly expensive. It starts with good architecture and a solid design. This reference architecture gives you a head start for creating your specific security and privacy design. You can use the proposed security and privacy principles and the requirements. Furthermore you can use or start with security models we present in this reference architecture as well. Also a list of example security system building blocks is presented. Since open source solutions can be valuable to lower security risks and reduce cost in your organization all presented solutions in this reference architecture are open source. This book also presents a list of criteria to evaluate the quality of OSS security/privacy solutions is.

Good privacy and security is difficult and complex. Making use of information presented in this book assures you do not have to reinvent the wheel so to say. Good security and privacy design for information systems is important. So do not lose your valuable time on trivial aspects. You have security problems to solve for your unique situation!

Good protection for our privacy is getting more and more difficult and expensive. In our opinion freedom requires very strong privacy protection assurances. We do not yet live a world where cyber security is always at a normal (low) risk level of protection to protect our core information assets like business and privacy related data. We still have a long way to go.

For privacy and security we need strong governance institutes that set rules to keep our (online) freedom.

If you want to help to remain freedom and want a more secure world, consider to support e.g. The Electronic Frontier Foundation (https://www.eff.org), a non-profit organization defending civil liberties in the digital world. Or support a similar local non-profit organization in your country.

Introduction

In our opinion security is a process, not a destination to arrive at. Good security design and implementation takes time, patience and hard work to achieve and maintain. You should always start with the basics by creating an architecture or overall design. As security and privacy will always be one of the most important subjects within IT the importance of good security and privacy will keep growing since companies will be even more depending on IT. Also the influence of IT will go deeper into our lives. Next to safety security and privacy will become more important when we realize the potential risks that come with new IT technologies.

This reference architecture is created to improve security and privacy designs in general. In our opinion it is time to stop reinventing the wheel when it comes down to creating architectures and designs for security and privacy solutions.

The reference architecture is not just another security book. Since libraries and book stores are filled with decent books on security and privacy we wanted to create a book that is all about reuse. There are two main pillars that drive this publication:

- Enabling reuse for companies of all sizes worldwide in order to design security and privacy solutions
- Creating an open reference architecture that enables collaboration and improvement in an easy way.

This reference architecture aims to enable you to create better and faster security and privacy solutions by reusing the content provided in this eBook. And to encourage collaboration on this eBook / reference architecture we created this reference architecture under an open license. We have chosen to use the Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) license. We know you like credit if you contribute to a publication. If you contribute you will of course be mentioned in all updates of this publication that follow. And since it is a true open license, your rights regarding this publication are no different than ours or other contributors.

To summarize this eBook is an open reference architecture aiming to help you to design better secure systems in less time and with less cost.

Why another reference architecture

Open publications for IT security and privacy are still rare. Despite the great work of the OWASP foundation many IT security organizations are not that open.

When you create a new medicine that can and will save millions of human lives it is not only ethical but also a moral right that people can use it. Science is there so we can build on each other's ideas.

This means progress for all. We all win. Within the field of software coding Open Source (OSS/FOSS) is becoming the new de facto model. Why create something anew that you already have created?

Within the field of security consultancy and security architecture Open is not (yet) the de facto standard. Of course some key assets as passwords or personal data should never be accessible. But creating security architectures and security designs is by many positioned as an art. That is strange of course. If you need a new color on your wall you do not call an artist, but a painter. The same goes for security: go for a proven open solution that has been used before. All solutions are of course mostly always context specific. No organization is the same. But that does not mean that every aspect for your architecture, and design should be new. We all use standard solutions where possible. Reuse of architecture and design is rare at the least. This reference architecture is aimed at enabling reuse of parts that are needed in every security architecture and design. That means less art, but the puzzle that remains is more interesting to solve. Since this is the real context related problem!

Availability of good references with solid reusable information makes creating security architectures easier and more fun. Easier because when you have a good security reference architecture you do not need various books to find out what already good proven parts for your architecture could be. More fun because you have more time to figure out what the best solution for your unique security challenge is. And we see thinking and resolving real security issues as fun. Minimizing security and privacy risks is always unique and context depended. E.g. unique stakeholders, different security control system (organization) and a different way of dealing with risks.

Why security and privacy

Privacy is getting more and more important. New technologies make our lives better but put our freedom and privacy under pressure. Terrorist and (cyber) criminals can be more easily detected by analyzing large amounts of data. Also 'diseases' can be better cured using more data of more people.

Currently great improvements come at a large price: Big data analytics systems are going over your user data and user data traces (e.g. mouse movements in web pages, location data) multiple times a day. Companies know better what you need, think and eat tomorrow than you. Your location is continuously being tracked, due to all the communication devices you use. Using public transport cannot be done anonymously anymore while cars are full of track and tracing technology.

When privacy is designed first just as security we should have less concern on security and privacy hacks. Also if more IT designs are open and published under an open license chances of mistakes in architecture and design will be less. Partly due to pressure of openness but also since more experts can contribute to lower security and privacy risks concerned with public or private systems. Of course: Transparency of governmental systems will be a (very) long way. Companies however see advantages of open solutions more and more. Using open solutions, open business models and open source software for IT. A large number of companies exist that benefit from using open designs along with open source software.

Many new technology companies are successful due to the fact that they promote open (FOSS) solutions. E.g. Companies like Automattic (https://automattic.com/), Acquia (https://www.acquia.com/) or IMatix (http://www.imatix.com/) are all very successful due to a true GPL OSS policy.

We know that privacy can be regarded as something totally different than security. This is why we had some resistance with combining a reference architecture for security with privacy 'things'. But our research showed that:

- Privacy has many relations with security. Many problems are similar.
- Privacy aspects are by far not yet taken serious into architectures and design the way they should be. It took decades and billion dollar (or euro) campaigns before security aspects were taken more seriously into account. And yet security is still difficult due the fact that doing it right gives no direct business value. Doing it wrong always means a true disaster for your business. And he same goes for privacy.
- Security and privacy are interrelated. Without security there is no privacy! Never.

Since privacy and security are very much interrelated both aspects will be outlined in this reference architecture.

Advantage of using this reference architecture

A good reference architecture saves time in many ways:

- You can create a solution architecture based on it for your specific situation.
- It enables you to speed up the process of creating a specific solution.

- It contains valuable content and general background information which can be used, reused or referred to.

Information security architecture is an abstraction of a design that identifies and describes where and how security controls are used. It also identifies and describes the location and sensitivity of both user and application data.

This open reference security architecture aims to help you create your context specific architecture faster and with higher quality.

This reference architecture is designed to assist and guide architects, security designers and developers to make better decisions and to reuse quality architecture knowledge regarding cyber security aspects.

The purpose of this document is to reuse good security principles, requirements and design patterns to save precious time and budgets. Since security by obscurity is in general not a good practice, we also provide a list of OSS security software products.

Systems built with tough privacy rules will not always guarantee that information including valuable privacy content is secure. Since security never is nor can be perfect a very secure system will always contain risks concerning privacy.

Who should use this reference architecture

The target audience for this reference architecture are security experts and companies who can see the benefit of reuse and using open source security building blocks. Specifically all business owners, security architects, security designers, asset owners, software developers, system administrators and (end) users who have a role in reducing security risks.

Scope of this reference architecture

Not all aspects of security and privacy can and should be outlined in a reference architecture. This reference architecture is not about teaching what security and privacy is. This reference architecture is not about providing detailed technical information on solutions that come across.

This reference architecture is also not a lecture book on how to design the perfect security solution architecture. There are many resources (books, courses, foundations) that will teach you the benefits of creating an (enterprise) architecture and how you can embed architecture into your agile way of working. Be aware of course that an agile way of

creating new products, systems or software gives some tension regarding security and privacy aspects. It is difficult to add security and privacy aspects at a later point if not done correctly from the start. So use new trends whenever possible. But if you were to design 'A human mission to Mars' important aspects like security and safety cannot wait till later.

Since you are reading this reference architecture, we assume you are already aware of the complex field of security and privacy. Very detailed books, papers and studies exist for learning what security and privacy really is. So this reference architecture will not give you in depth detailed background information on all security and privacy aspects. Not from an organization point of view and certainly not from an IT point of view.

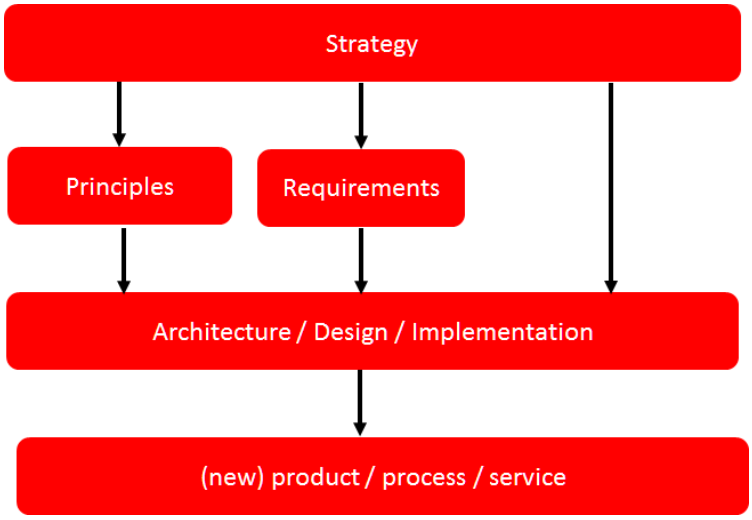
The scope and aim of this open security architecture is to enable you to create better and faster security solution architectures and designs using open reusable building blocks and standards. Within the scope of this reference architecture are:

- Security solution aspects, e.g. models, that must, should or could be reused in a security or privacy solution architecture.
- Information that can be reused in an easy way in your context specific security/privacy solutions. E.g. security and privacy principles.
- Criteria aspects you can reuse when selecting security solutions for your solution architecture.
- (Sample) Security/Privacy Solution Building Blocks that are created for reuse. These SBB's serve as example to give you a more in depth overview of possibilities you are maybe not familiar with.

Creating security architecture consists of the following high level steps:

- Dive in the business strategy and organization;
- Gather security and privacy principles and requirements;
- Determine important constraints that apply to your architecture or design. There are always constraints, e.g. time, budget, subject matter experts available etc.
- Derive the architecture building blocks from your architecture or design. Architecture building blocks help you to scope your solution. Using architecture building blocks gives a clear view on (new) integration aspects and where completely new solutions fit in the total IT landscape.

- Select (or create, buy) the new Solution Building Blocks. Prerequisite is of course that the functionality and technical constrains must be clear. Often prerequisites are derived from the previous design step.

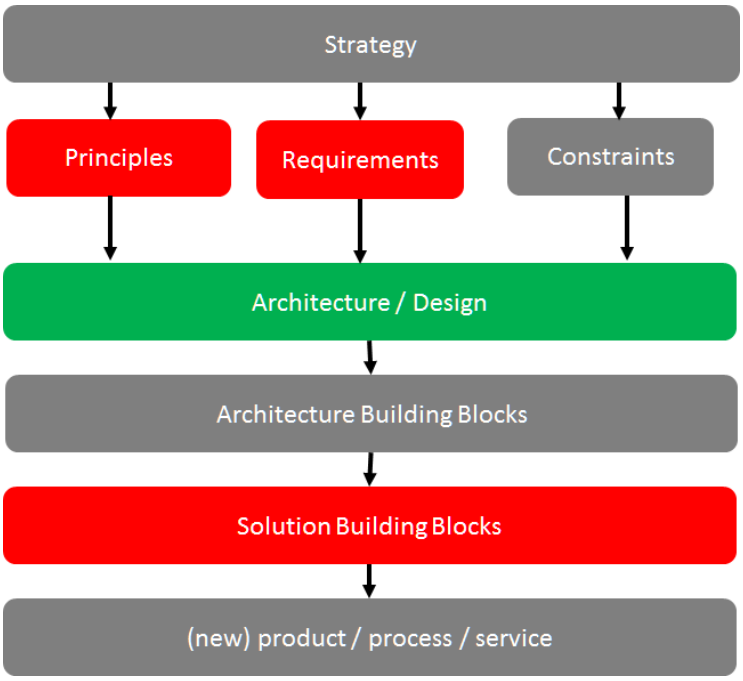


Within this reference architecture we will focus on the following subjects that you should face when creating a security or privacy solution:

- Principles: We will provide a reusable list of security and privacy principles. Since this open security and privacy reference architecture has an Open approach we encourage you to add your principles to the open data source we created to help others from reinventing the wheel again and by doing so they save time.

- Solution Building Blocks: We provide a list of solid reusable security and privacy tools and building blocks. Of course all tools and building blocks are open source. One core principle is that good security should be open. Within this eBook a detailed outline is given on the question if extra risks factors are involved in using open source solutions.

- Reusable architecture and design patterns for security and privacy problems. During the architecture and design phase threat models are constructed. This document contains generic threat models, since these are reusable. That can be improved when the model is made publically available.



So many aspects regarding security and privacy our not in scope of this reference architecture.

What about security patterns?

In system design, coding and architecture you should strive to reuse predefined patterns. A pattern is a reusable way to solve a standardized problem. This can be in software code, design or an organization problem.

Good patterns within the security and privacy field are rare. We did research on available reusable patterns that can help creating security or privacy solutions faster. Our findings are:

- Good described reusable security and privacy solution patterns are rare.

- Reusable architecture and design patterns for security and privacy problems are scarce. Most relevant patterns are vendor specific, so are targeted to the solution building block reuse aspects.

- Use of patterns can increase complexity. Understanding pattern language and semantics is important before being able to judge if your chosen pattern applies to the unique challenge that must be solved. Since libraries are written on generic problem solving methods (note: the golden book is still not found) some precaution using patterns is very healthy!

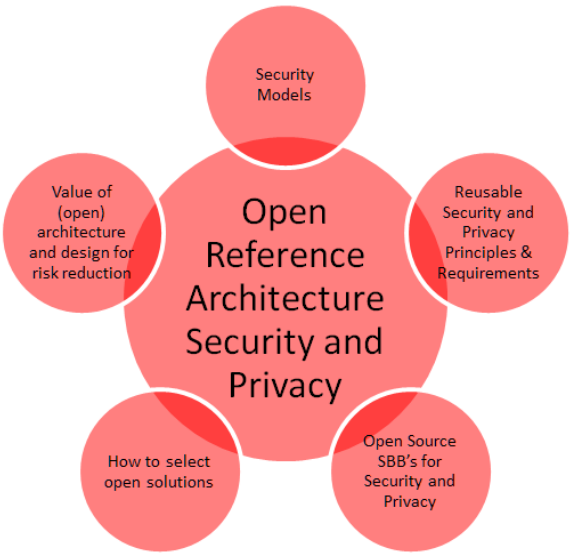
- Developing patterns (also in a collaborative way) for a reference architecture takes up a lot of precious time while the practical use (or reuse) in a solution architecture is always questionable.

We hope good developed patterns for dealing with typical security and privacy problems will be developed in future. Also we hope these patterns will be developed in an open collaborative way and published under an open license so everyone can benefit and participate. Some good attempts have been done, so maybe time for a new OWASP project to give it a boost.

Currently we think that when you write a good solution architecture in which you describe your problem clearly will help to create a library of reusable solution patterns for security and privacy. One import constraint is that your solution architecture should be published under an open license somewhere on the internet. In this way every organization, security designer can benefit. Some governments already publish their architecture documents under an open license (CC) on the internet. This is a great way for governments to align better with society. Everyone can see how complex digital information systems become and can suggest improvements. Detailed configuration information is not needed to judge the risks of security or privacy vulnerabilities. Companies worldwide are still very anxious to benefit from the possibilities that a more open transparent company (using open licensing) can give.

How this reference architecture is structured

This reference architecture is built around information that helps you creating security or privacy solution architectures.



It is also built to give you reusable information in an easy to find way. The next chapter ('Security Models') deals with models, attack vectors and information that helps you create the threat model you need to develop in your solution architecture.

The chapter 'Security and Privacy Principles' presents solid security and privacy principles. Focus is on use and reuse. The chapter 'Using Open Source for security and privacy protection' outlines facts to demystify common fads regarding use of Open Source and security and privacy products. This chapter outlines how to evaluate OSS Solution Building Blocks for security and privacy applications. The chapter 'Open Source Security and Privacy products' presents a list of great OSS solutions available to be incorporated into your security or privacy solution or just to take a look at.

The appendixes will give you information on reference used, as well as information on how you can contribute with the next version of this reference architecture.

Security Models

Introduction

The essence of information security is to protect information. It is just that simple. So whenever possible do not make it more complicated than needed. Complexity for cyber security and privacy arise when information needs to be shared or must be made accessible by some digital device. The world where information was only available in physical archives is long gone. The focus from physical information security is shifted to cyber information security. But be aware: Crucial principles of centuries of physical information protection are still valuable today. Especially principles related to the intangible soft issues when information is shared. Ever wondered how some organizations managed to keep their valuable information secret for many decades?

Information protection is needed against unauthorized access, use, disclosure, modification or destruction. That means several security measures are needed to protect information from unauthorized viewers. Measures can be implemented by procedural, physical or with complex IT tools. But before classifying and creating or finding good measures it is essential that the problem field is made clear.

Creating effective solutions for information security problems can be done by creating a model of the problem situation. Within a model all elements that relate with the problem situation are brought together to study effective solutions. Without going into detail on system science or problem solving theory: in general systems consist of sub-systems, objects, functions or processes, and activities or tasks.

The key in creating a good model to solve a specific information security problem is to model the problem, not the complete system with all elements. This because modelling the world completely is ineffective, time consuming and it does not give a direct answer to solve a problem situation. It is far better to start with a small model of a problem and create extensions on this model if needed.

The field of modelling problem situations to solve information security problems is not new. Many models in literature exist. Reusing a good model can save you time and safeguards you from making mistakes. A prerequisite is that you start with a good model that can be trusted and is intensively reviewed by large numbers of subject matter experts.

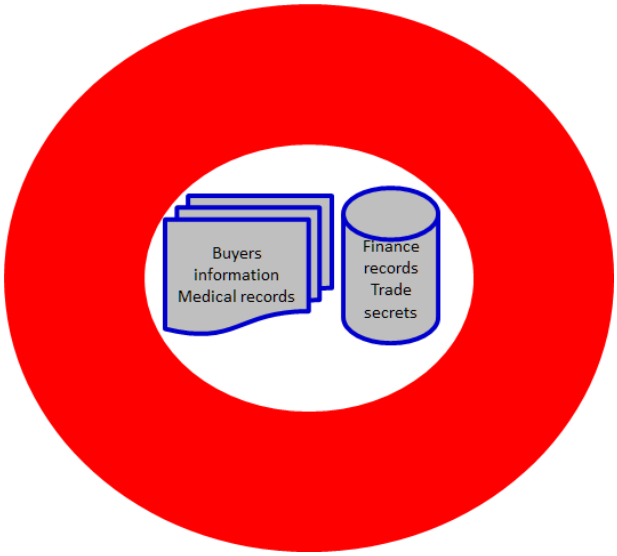
There are many good security models that can assist in creating a solution architecture to solve a specific security problem for an organization. Mind that a model can be expressed in many different forms. E.g.:

- One or more images;

- Text;
- Software model

Within the field of modelling a distinction can be made between ‘hard’ and ‘soft’ models. Hard models are often mathematical (risk) models whereas soft models are more quality based models. Since using hard models often gives a false sense of reliability and requires full insight of all assumptions made it is more productive to reuse soft security and privacy models. When creating solution architecture, you need:

- A threat model (what are the threats your solution gives protection against)
- Insight in commonly used attack vectors. This means you need to have some view on the attack vectors used in the use case?



Creating a good security or privacy design or architecture means you never ever start with selecting tools for solving your problem! Selecting tools should be the last phase of your security or privacy design phase. You select tools when it is clear that the tool will support in solving your security or privacy problem. Tools alone are never enough to solve security or privacy problems. You need to fit in tools within your security and privacy processes. Several problems exist with many IT security tools that will hit you when you start too soon with the solutions instead of a thorough problem diagnosis and solution design. Wrongly selected security and privacy tools give the following issues:

- High costs;
- Complex challenges to implement and manage;

- Daily administration of a chosen tool set requires significant IT effort while it remains unclear if the tools are effective in reducing security risk;
- Overlap in functionality of security application landscape. More is not always better. To be able to justify the application of security tools for your problem a context specific security architecture should give input to the following questions:
    - What is protected with what?
    - What are the main threats we need protection against?
    - What is not protected by information security policies or tools?
    - What is in scope or out of scope for your security architecture? E.g. business continuity management, safety management, financial risk management, daily IT operations, physical (building) security etc. In the end everything has a relation with information security, but you cannot cover all business aspects using an information security architecture document. The key is to focus and keep the scope clear or else complexity will become overwhelming.
    - What architecture or design decisions have been made and must be validated explicitly?
    - What is the model of your protection? It is far more easy to evaluate and improve a model, than adding new or improved security products continuously. Make sure that within operational security management processes learning and improving are key periodic targets.
    - Does the security model cover all crucial security and privacy principles and requirements?
    - Are the residual risks when this solution acceptable for the key stakeholders?

IT security in general is seen as a complex problem field, due to the many technical and nontechnical aspects involved. Since 100% information security is impossible, being able to qualify risks is crucial in getting an accepted level of security protection. Good modelling helps you to qualify security and privacy risks.

In general, it is far more easy to reuse proven concepts and models when creating your own security model. This way you build on the work of others and using a good model reference will reduce the risk of making crucial mistakes.

This section covers some commonly used models and elements that can be reused when creating a solution for a specific information security problem.

Elements that are presented are attack vectors, some examples of security personas and some great security models that can assist you when creating your security design.

Common attack vectors

Good security is goal oriented. Good security architecture is tailored to your situation. When defining a product or new (IT) service one of the key activities is to define your specific security requirements. Defining requirements is known to be hard, time consuming and complex. Especially when you have iterative development cycles and you do not have a clear defined view of your final product or service that is to be created.

Defining attack vectors within your security requirements documentation is proven to be helpful from the start. Attack vectors will give more focus on expected threats so you can start developing security measures that really matter in your situation from the start.

Attack vectors are routes or methods used to get into information systems. Attacks are the techniques that attackers use to exploit the vulnerabilities in applications. Many attack vectors take advantage of the human element in the system or one of the maintenance activities defined for the system, because that's often defined as the weakest link.

Within the IT cyber security world many terms and definitions are used. Attack vectors usually require detailed knowledge to judge whether the vector is relevant in a specific situation.

Some attack vectors apply to critical infrastructure components, like NTP or DNS. E.g. in a rogue master attack, an attacker causes other nodes in the network to believe it is a legitimate master. Contrary to spoofing attacks in the Rogue Master attack the attacker

does not fake its identity, but rather manipulates the master election process using malicious control packets.

The good news is: The number of possible attack vectors is limited. The bad news is: The ways an attack vector can be exploited is endless. Unless decent security measures are taken to minimize attacks using this specific attack vector. Good designed security solutions are not that complicated and complex after all.

Common attack vectors are:

- Analysis of vulnerabilities in compiled software without source code
- Anti-forensic techniques
- Automated probes and scans
- Automated widespread attacks
- Client validation in AJAX routines
- Cross-site scripting in AJAX
- Cryptographic Performance Attacks
- Cyber-threats & bullying (not illegal in all jurisdictions)
- DoS Attacks
- Email propagation of malicious code
- Executable code attacks (against browsers)
- Exploiting Vulnerabilities
- GUI intrusion tools
- Industrial espionage
- Internet social engineering attacks
- Malicious AJAX code execution
- Network sniffers
- Packet Manipulation

- Packet spoofing
- Parameter manipulation with SOAP
- Replay Attack
- RIA thick client binary vector
- Rogue Master Attack
- RSS Atom Injection
- Session-hijacking
- Sophisticated botnet command and control attacks
- Spoofing
- Stealth and other advanced scanning techniques
- Targeting of specific users
- Web service routing issues
- Wide-scale trojan distribution
- Wide-scale use of worms
- Widespread attacks on DNS infrastructure
- Widespread attacks using NNTP to distribute attack
- Widespread, distributed denial-of-service attacks
- Windows-based remote access trojans (Back Orifice)
- WSDL scanning and enumeration
- XML Poisoning
- XPATH injection in SOAP message

It is recommended that you specify in your solution architecture the attack vectors that apply to your use case. Remember to put the explanation of the attack vectors used in an appendix, since not all your stakeholders will know what e.g. 'Spoofing' is.

Hosting, hardware, firmware and other invisible threats

Computer security has become much harder to manage in recent years. This is due to the fact that attackers continuously come up with new and more effective ways to attack our systems. But also the emerging trend of Cloud Computing created an extra level of complexity within the field of cyber security and privacy protection.

A commonly wide spread fad is that Cloud Hosting is more secure than on premise. The truth is that it is different. Security principles and all attack vectors still apply. The main factors that make Cloud hosting more complex to manage are:

- Less control
- Technical insight in exact physical and IT security measures are often unknown.
- Influence and control on continuous operational changes on the cloud hosting facilities are not transparent for cloud consumers.
- Trust plays a great role. You must have trust in audit and security reports created by a third party. The advice is to obtain always a right to perform a security audit yourself, but at large cloud hosting providers this is often not allowed.

Whether you use Cloud hosting of host your computer services still on your own data centre all hardware threads still apply.

Since true open source hardware is still seldom seen, currently your valuable information is vulnerable due to the following more hardware related attack vectors:

- BIOS attacks. BIOS is always written to a non-volatile storage device such as an EEPROM
- Firmware attacks
- Physical device tempering. Mostly done by rewiring CPU's, CPU boards. Famous are of course the attacks on Crypto Devices (HSM's) but since hardware tempering on normal hardware is so easy you seldom hear how easy hacking on 'standard' computer hardware devices is.

- Physical data centres. Your data is not (never) secure in a cloud you do not control or manage.

An attack vector that many people forget to consider is the boot process itself which is almost completely controlled by the BIOS.

When you are still in control of your own computer hardware, consider to overcome the malicious attacks on BIOS by one the following methods:

- Digital Authentication Method
- Rollback Prevention Method
- Physical Authentication Method

Threads related to hardware are often invisible. This does not mean they don't exist. Since computer hardware is seldom open, many threads are still not widely known. In order to protect your core information you should always take measures to be able to reduce the likelihood of getting targeted by attack vectors that are hardware related. Many examples exist of poor designed CPU's, firmware, network devices, storage devices etc. with offers great opportunities to attackers.

Security Personas

Humans are the most important threat to security and privacy.

One of the tools of IT architects and UX designers is to work with so called 'Personas'. Personas are fictional characters created to represent the different user types that might use a system, website, product or service. Using personas is common practice when dealing with UX design. But when developing a security architecture for a new system, service or website security personas are also valuable to use. Security Personas force you to think different about the goals and behaviour of attackers that are going to hit your system.

Security Personas identify the user motivations, expectations and goals responsible for driving bad behaviour. Of course not all personas will behave bad on purpose. Sometimes mistakes on the use of the system or social engineering will affect the way a persona can compromise your system.

Benefits of Personas

Personas help to focus and help to make design decisions concerning IT components by adding a layer of real-world consideration to the conversation. They also offer a quick and inexpensive way to test and prioritize those features throughout the development process. In addition, they can help:

- Stakeholders and management to discuss architecture building blocks to protect your system.
- Information architects develop informed secure wire-frames knowing possible interface behaviour.
- System security engineers/developers to decide which approaches to take based on user behaviours.
- Testing

For security personas it is good to outline:

- Demographics such as age, education, ethnicity, and family status.
- The goals and tasks they are trying to complete using the system (or website),
- Their physical, social, and technological environment.
- Responsibilities: As implemented in future Identity and access management system, but also the formal organization responsibilities belong to the role within the organization.

Defining security personas is not hard. Some examples of security personas:

- Employee
- Visitor (in person)
- Internet visitor (web)
- Administrator
- Manager
- Director/CEO
- Angry customer
- Competitor/rival
- Neighbours

Use security personas in your security architecture so the proposed security measures can be designed more in depth and evaluated since the security personas are part of your

security model. The list given in this section can be used as starting point to expand the personas for your context more in depth.

Threat Models

This section is not about teaching you how to model you specific security or privacy solutions. By now you know that your model should be built out of attack vectors, security personas and security and privacy principles and requirements. The next chapter of this reference architecture deals with reusable principles in depth. First we present valuable models that can be reused when created a security or privacy solution architecture.

Security threat modelling, or threat modelling, is a process of assessing and documenting a system's security risks. Security threat modelling enables you to understand a system's threat profile by examining it through the eyes of your potential attackers. Your security threat modelling efforts also enable your team to justify security features within a system, or security practices for using the system, to protect your corporate assets.

Many ways exist to build a threat model but in essence a threat model is a conceptual model that:

- helps to understand a situation and
- is helpful in reducing security or privacy concerns. So helpful in solving your security problem.

A security or privacy conceptual threat model is usually built of relevant elements and their relations that matter in a security problem situation.

In general, a conceptual model is constructed based on a specific problem situation you want to solve. In our case the aim is to outline important concepts regarding security and privacy. So our collection of conceptual models is aimed at generic reuse.

Since the real-world problems of security and privacy are outlined in a large number of publications, within this section we only present conceptual models that are based on the following selection criteria:

- Generic use;
- Non-commercial;
- Open.

With open we mean that the institute or company created the model has an open process that allows everyone to improve the model. Of course open is not always really open

without borders and thresholds. Even the open group is not really open for public participation, since large memberships fees form a threshold. The OWASP foundation is however one of the best examples on how open should be. That is open license on content (common creative) and no impediments and no requirements for participants who want to join the working groups.

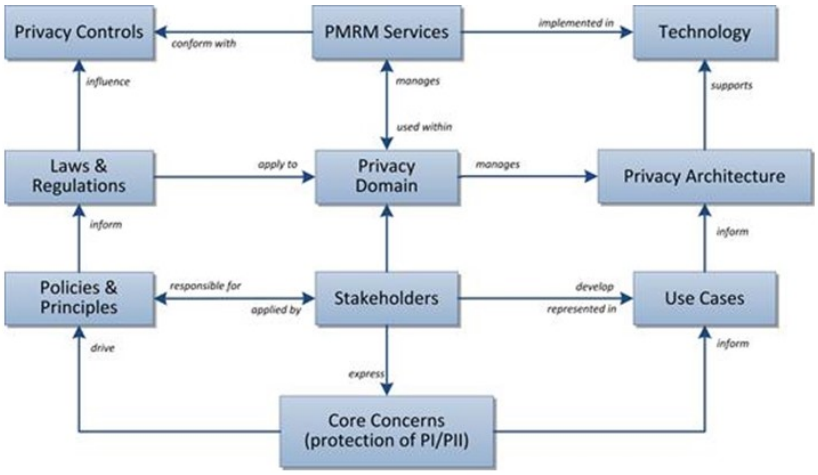
For security and privacy many models exist. Most models are aimed for evaluating risks for auditors and other stakeholders. In the sections below a collection of (almost open) security and privacy models.

Privacy Management Reference Model

The Privacy Management Reference Model and Methodology (PMRM) of the OASIS group provides a model and a methodology for:

- Understanding and analysing privacy policies and their privacy management requirements in defined use cases; and
- selecting the technical services which must be implemented to support privacy controls.

The model is particularly relevant to evaluate use cases in which personal information (PI) flows across regulatory, policy, jurisdictional, and system boundaries.



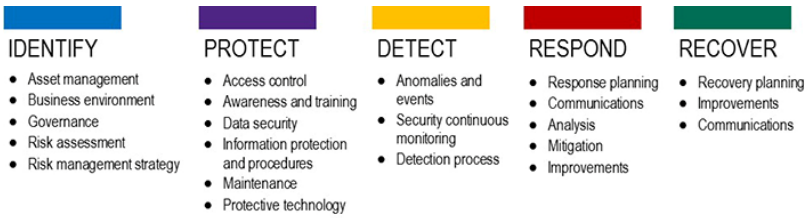
Source: OASIS - http://docs.oasis-open.org/pmrm/PMRM/v1.0/csd01/PMRM-v1.0-csd01.html

More in-depth information regarding this model can be found on the OASIS site (see references).

NIST Security framework

Whenever you feel the need to draw a process regarding security or risk processes: resist the temptation! The US based NIST organization is a well-known governmental organization that offers great publications on all thinkable subjects regarding security.

One of the simplest, yet most frequently model is displayed here below.



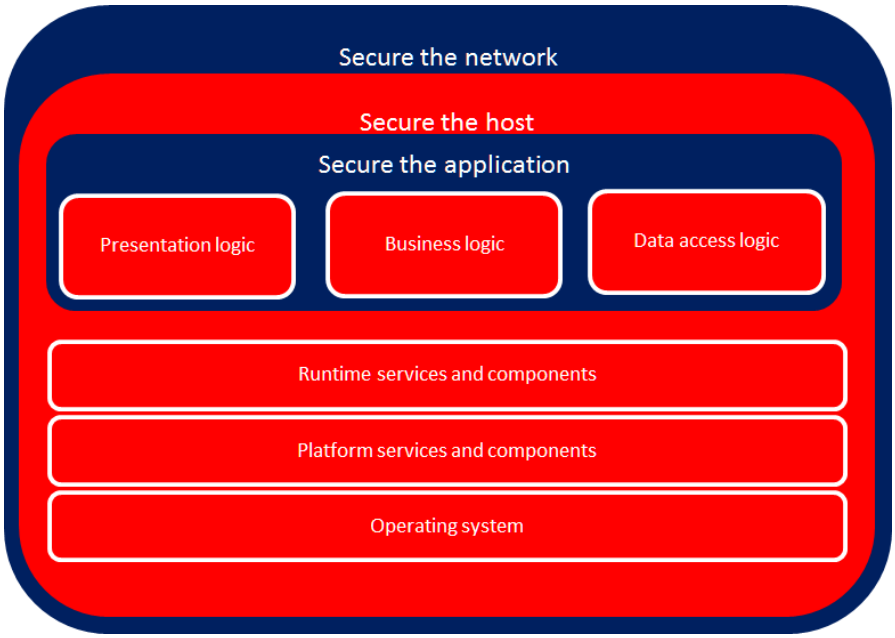
On the NIST site (see references) you can find in-depth information regarding all sub functions of this security framework. The experience is, is that it is far better to check what in your use case needs special attention. If you ever feel the need to create your own security framework, think again. In essence all come down to the high level framework described by the NIST organization. Using a broad used security framework has a number of advantages:

- Easier communication with stakeholders;
- Easier knowledge and experience transfer between security experts of different organization;
- Saves time, time you can use to solve the real context specific issues regarding practice use and implementation of the security functions.

Jericho Security Model

The Jericho(tm) Security architecture model is built upon principles. The advantages of using the Jericho model for security are:

- A security architecture model built upon the Jericho conceptual model is built around maintaining flexibility and protects the most important security objects for the stakeholders.
- Integration: Easier to build secure processes with other companies and trusted partners.
- Simplifies use of public networks and cloud solutions
- Aimed for use of open principles and open solution building blocks.

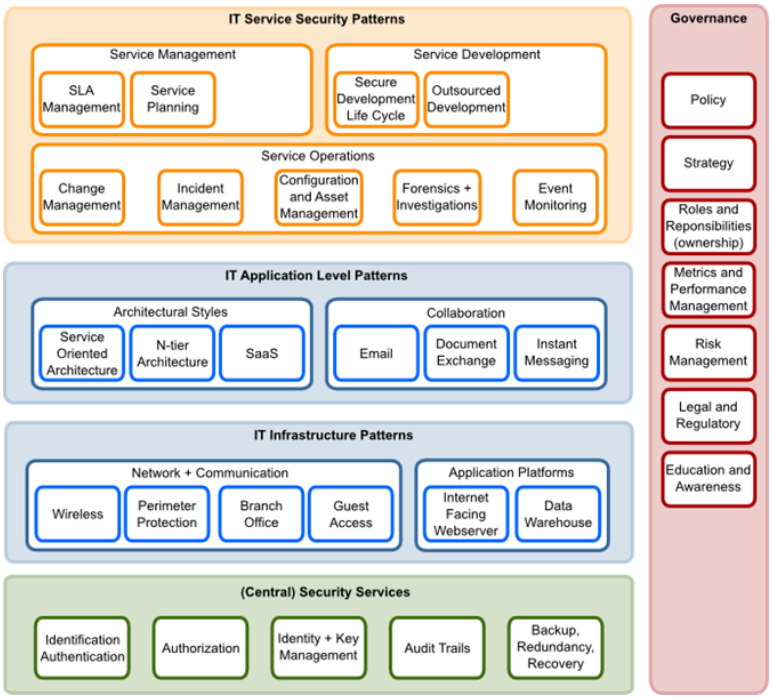


Unfortunate the Jericho framework is not a real open security framework. It is copyrighted by the open group (see references chapter for more information on this model). There are trademarks involved and all publications are copyrighted. However due to the work of many we can make use of the developed knowledge within the Jericho working group. The Jericho Forum®, a forum of The Open Group, was formed in January 2004 and is no longer active. However, the approach of this forum towards security is still alive.

Security Architecture Landscape (OSA)

Thanks to the Open Security Architecture (OSA) group there is a real open security landscape (http://www.opensecurityarchitecture.org/). All OSA material is CC by sa licensed, which means you can freely use and improve it.

Below is the OSA Security architecture landscape:



Source: OSA (http://www.opensecurityarchitecture.org)

The OSA Security architecture is based on patterns. Which mean for every pattern defined the aim of the community was/is to develop a standardized solution description. Unfortunate the OSA community is not very active anymore, so all IT security patterns around cloud are not yet incorporated.

For a number of reasons we have chosen not to use patterns in this security and privacy reference architecture. However in some cases using patterns can give an advantage. (See the Introduction, section 'What about security patterns?' for more information).

Software Assurance Maturity Model (SAMM)

The Software Assurance Maturity Model (SAMM) is an open framework to help organizations formulate and implement a strategy for software security that is tailored to

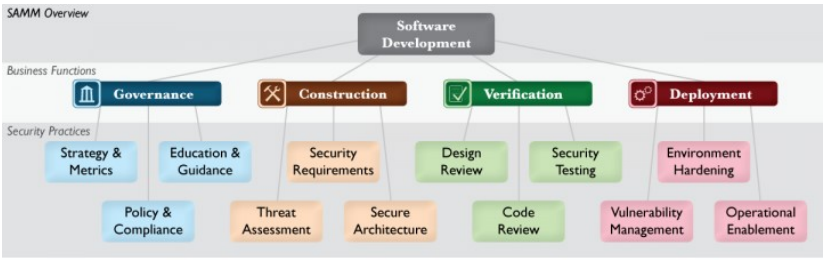
the specific risks facing the organization. SAMM is useful resource if you are working on a process architecture that is needed to control all kind of aspects of software security. Our advice is to take the processes as defined in SAMM as point of departure within your security process design documentation. Formulating processes yourself in not productive, so use this valuable source of information instead of reinventing the wheel.

To get the baseline situation of your security process architecture fast in scope, you can use a SAMM self-assessment test (see APPENDIX). Using a self-assessment test you can get a very quick overview on the status of the IT security processes within your organization. SAMM is an OWASP project.

SAMM will aid in:

- Evaluating an organization's existing software security practices
- Building a balanced software security assurance program in well-defined iterations
- Demonstrating concrete improvements to a security assurance program
- Defining and measuring security-related activities throughout an organization

As an open project, SAMM content shall always remain vendor-neutral and freely available for all to use.



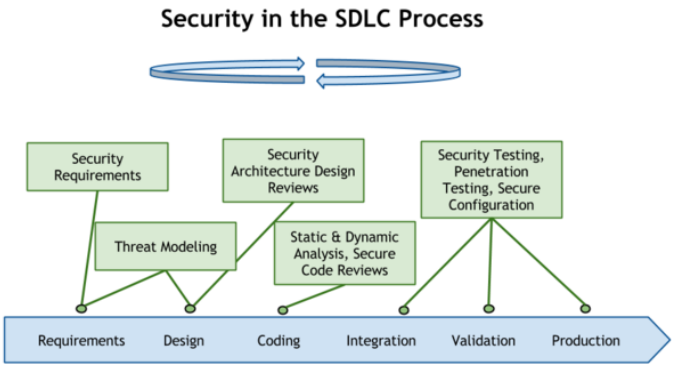
Source: OWASP

Reuse of the SAMM process and usage should be encouraged. This OWASP project is like all OWASP projects a real open project. All content is available under a Creative Commons License (by-sa). If you want to improve this SAMM framework, OWASP is a real open foundation where everyone can participate without borders. Also all communication and collaboration is truly open.

The SAMM model was first aimed at evaluating the status of software security within an organization. However due to the use in practice the framework can also be used to improve many other aspects surrounding security and privacy.

Security within the SDLC process

The view below (source OWASP) is a model of how security fits into the SDLC (Software Development and Lifecycle) process. Within almost every solution architecture you should take the SDLC into account to position where your solution fits and how maintenance is positioned within the SDLC phases.



Source:OWASP (https://www.owasp.org/index.php/CISO_AppSec_Guide:_Application_Security_Program)

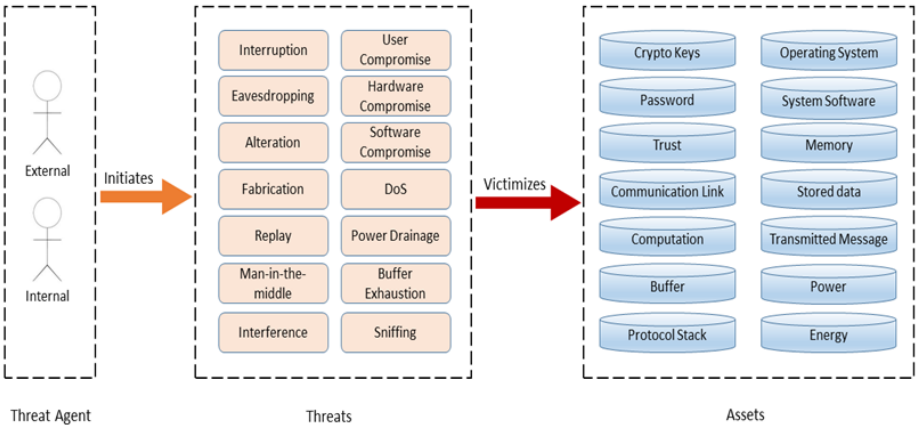
Security and privacy should be embedded in the SDLC process. Always. The OWASP conceptual model of the (simplified) SDLC chain shows on high level where security activities hit the SDLC process.

IoT Threat Model

We should be happy: The IoT (Internet of Things) is not everywhere present yet. When IoT is migrated from fiction to reality, security and privacy will be under enormous risks.

Internet-of-Things is a result of a technical revolution, which reflects with future computing and communications including existing and evolving internet. Over the time Internet technologies have evolved, and become Internet of Things. With the advent of this paradigm the dream to convergence everything, and everyone under a single umbrella has come true. Machine-to-machine (M2M), Radio Frequency Identification (RFID), context-aware computing, wearables, ubiquitous computing, and web-of-things all are considered to be seamlessly integrated into a global information network, which has the self configuring capabilities based on standard and inter-operable communication protocols .

Below a generic threat model for the IoT world:



Source: http://secret.cis.uab.edu/research/iot-security/

Note the view is not complete. Missing these views are:

- IDS, pentest tools, correlation tools etc (or under system security)

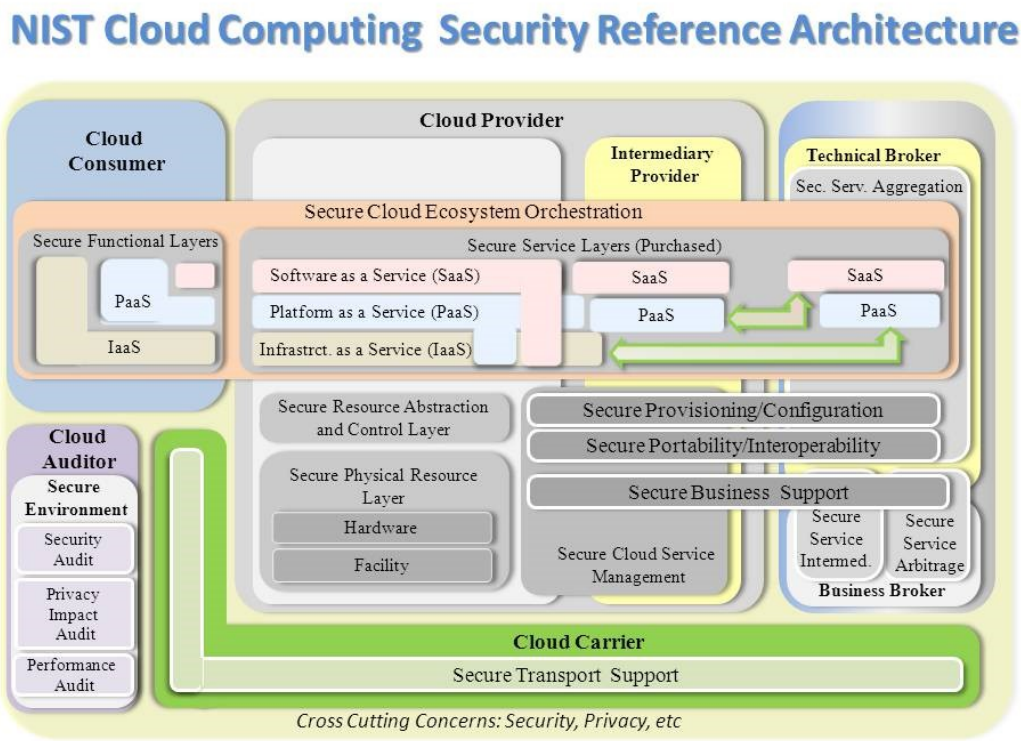
This IoT thread model and views are good for addressing the following areas in more detail in your security solution:

- Confidentiality
- Integrity
- Availability
- User Management
- Network Security
- Key Management
- Security Management
- Governance
- Risk
- Regulation

- Audit
- Access Control
- Standards for Interoperability

NIST Cloud Computing Security model

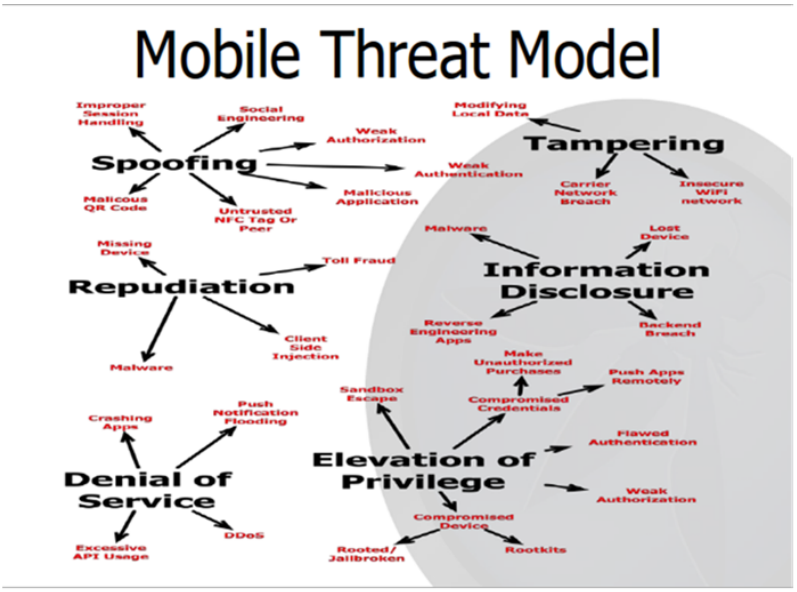
Sooner or later you will be creating a solution or privacy architecture where cloud hosting plays a significant part. The NIST cloud computing security reference model is a very good model to use as reference.



Mobile Threat model

Since mobile is everywhere, you should always take mobile threats serious in your solution architecture. Even if you think you have a special gateway for mobile traffic, most devices are always vulnerable for mobile threads.

The model presented here below can help in identifying the threads.



Source: http://pokerfuse.com/site_media/media/uploads/features/mobilethreatmodel.png

DDoS model

DDoS attacks are hard to prevent. However, every security or privacy architecture should take DDoS attacks into account. This to design solution that are more resistant against the easy DDoS attacks.

Problems due to DDoS Attacks:

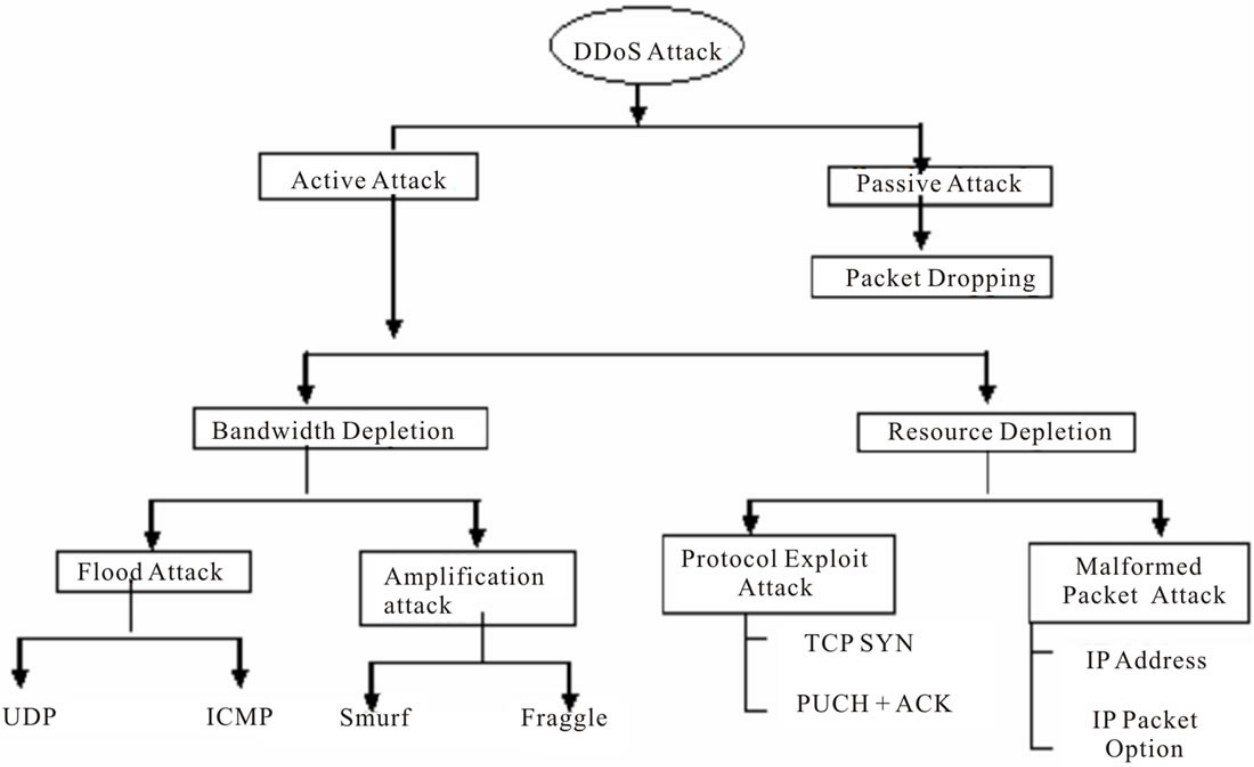
• DDoS attack is an attempt to make a systems inaccessible to its legitimate users.

• The bandwidth of the Internet and a LAN may be consumed unwontedly by DDoS, by which not only the intended computer, but also the entire network suffers.

• Slow network performance (opening files or accessing web sites) due to DDoS attacks.

• Unavailability and inability to access a particular web site due to DDoS attacks.

The model below gives a DDoS attack taxonomy. This can be useful if you are designing solutions to be more resilient against DDoS attacks.

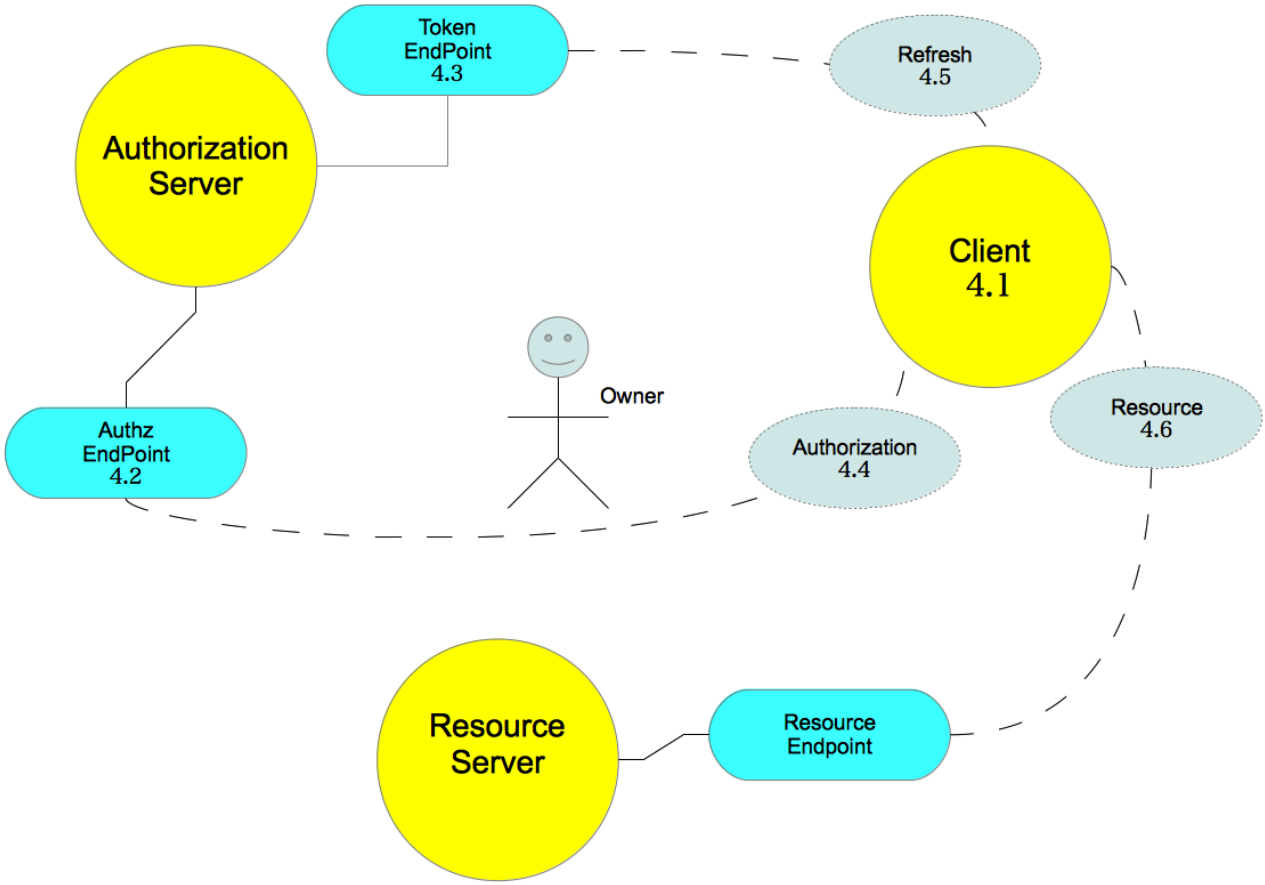


REF: http://file.scirp.org/Html/5-7800164_34631.htm

OAuth 2.0 Threat Model

Using the OAuth protocol gives you many advantages. And since this protocol is open you can save a lot of time when making use of the OAuth Threat Model when using OAuth in your use case. A detailed description of the thread model is found in RFC 6819 (http://tools.ietf.org/html/rfc6819).

In the picture below the visual of the threat model, where the numbers are references to the section in the IETF RFC.



OAuth 2.0 threat model.

(source: http://hdknr.github.io/docs/identity/oauth_threat.html)

Security and Privacy Principles

Every organization is different. However, when you are faced with the challenge to create a new (IT) product or service having good principles requirements before you start will help. Always.

We have simplified this complex but crucial step needed in every project. In this chapter you find lists of:

- Security principles and
- Privacy principles

We encourage reuse! We also encourage you to add principles or correct these principles. In time we are aiming to create a collection of the best e.g. 100 principles for security and privacy that can be used when creating a specific solution architecture. A good reference architecture should save you time when creating a solution architecture, so use or reuse these principles from this architecture. In this way you have more time to focus on the specific context related problems. In essence the use or reuse of good security and privacy principles prevent you from making crucial design and implementation mistakes in your use case.

What are principles?

Principles are statements of direction that govern selections and implementations. That is, principles provide a foundation for decision making.

Principles are used within business design and successful IT projects.

Definition:

A principle is a qualitative statement of intent that should be met by the architecture.

Security architecture principles are used to translate selected alternatives into basic ideas, standards, and guidelines for simplifying and organizing the construction, operation, and evolution of systems.

It is important to draw an early differentiation between standards, requirements, and principles.

- Standards are “musts”; that is, they require compliance.
- Requirements articulate specific needs that must be met by a specific solution.
- Principles, on the other hand, are more general and serve as a framework for making choices by providing guidance about the preferred outcome of a decision in a given context.

As such, the purpose of our collected principles is to support decision making with regard to security and privacy design within all organizations.

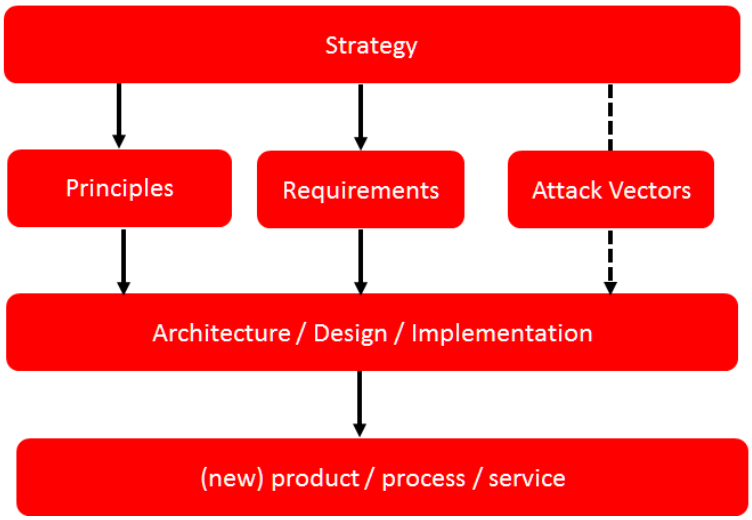
The following criteria can be used to determine the quality of a principles:

- Understandable: Every stakeholder involved should be able to understand the meaning, purpose and implications of a principle.
- Consistent with other defined (or selected) principles.
- Aimed to the goal.
- Usable.

Principles will guide architects, consultants and designers with decision making. Within business design and architecture, you find many people with strong opinions with what a good and usable principle is or is not. Discussion is always good to get a better understanding of each other mental maps. However, discussions on what a good security principle is should be target on what you can do with principles. How will principles help you and your company? Can principles help you doing projects faster and better? Can principles prevent your company architecture and software systems becoming the next IT over complexity landscape?

Having security and privacy principles are a crucial foundation as they establish the basis for a set of rules and behaviours for any organization.

Security Principles are statements of direction that govern selections and implementations. Security principles provide a foundation for decision making and are a fundament for the project.



Principles or requirements?

The exact difference between what a principle is and what a requirement is, is a long running debate. Long running debates does not make your organization more secure. It is time consuming and in the end no one is right. So do not fall in the trap of such a semantic discussion.

Security and privacy principles have the following characteristics:

- Principles are general rules and guidelines.
- Principle are often a qualitative statement of intent that should be met by the architecture.
- Principles are guidance to help making decisions with the help of rules.

Your security and privacy design should be created based upon many design decisions. Using (approved) principles will help.

Security and privacy requirements tend to have the following characteristics:

- Can be SMART (https://en.wikipedia.org/wiki/SMART_criteria) formulated. So you can test if a requirement is implemented well.

- A requirement is more context specific than a principle. E.g. your users can have different requirements on user-friendly and secure login than users of another company.

- Requirements can be prioritized within a project, where principles are more directly shaping an architecture or design.

Principles can be regarded and threated as requirement, but due to the formulation requirements seldom can be directly used as generic principle.

Although the difference between security and privacy principles and requirements is most of the time hard to make, having requirements in addition to principles will improve a privacy or security design.

This because using requirements leaves more room to discussion and prioritization with direct stakeholders.

What are requirements?

Many studies show that poor requirements are a prime cause of project failure or insufficiency. Tools that assist you with creating good security requirements or let you reuse security requirements are rare. But it is crucial for good security that you start with collecting principles and requirements before coding or buying software.

Within traditional waterfall methodologies a requirements document is created by a business analysts and subject matter experts who would spend significant time on creating requirements that are never complete. Developers are often faced with challenges deadlines and have little time to handle and implement all requirements correctly. In practice there is simply have no time to get familiar with the real meaning and purpose of all requirements and developers make guesses on the real goal of requirement statements.

In most projects today, a lapse of several months would either invalidate these requirements or miss the market window altogether. Internet speed and agility mean that projects must be quick to market and must evolve continuously to meet the changing needs and demands of their users.

Common Mistakes regarding security and privacy requirements

- Basing a solution on complex or cutting edge technology and then discovering that it cannot easily be rolled into the 'real world'.

- Not prioritising the User Requirements, for example 'must have', 'should have', 'could have' and 'would have,' known as the MoSCoW principle.

- Not enough consultation with real users and practitioners.
- Solving the 'problem' before you know what it is.
- Lacking a clear understanding and making assumptions rather than asking.

Requirements gathering is an essential part of any project and project management. Understanding fully what a project will deliver is critical to its success. This may sound like common sense, but surprisingly it's an area that is often given far too little attention.

Many projects start with the barest headline list of requirements, only to find later the customers' needs have not been properly understood.

Since security and is always in the end risk based we recommend that you prioritise your chosen requirements. We advise to use the de-facto standard: the acronym MoSCoW.

This stands for:

- M – MUST: have this.
- S – SHOULD: have this if at all possible.
- C – COULD: have this if it does not affect anything else.
- W - WON'T: have this not now, but would like this in the future.

Requirements marked as "Won't" are potentially as important as the "Must" category. Classifying something as "Won't" acknowledges that it is important, but can be left for a future release. In fact a great deal of time might be spent in trying to produce a good "Won't" list. This has three important advantages:

1. Stakeholders/Users do not have to fight to get something onto a requirements list.
2. Thinking about what will be required later, affects what is asked for now.
3. The designers seeing the future trend can produce solutions that can accommodate these requirements in a future release.

Reuse of requirements provides a number of benefits, including the following:

1. Motivation for selection of components: Requirements guide the selection of optimal components for reuse. When requirements are transferred between development efforts, the rationale behind the original component selection decision is made available to the system designer.

2. Context for reuse decisions: Requirements trace back to information gathered from domain experts and system users. Requirement-based reuse decisions are set in the context of domain processes or specific implementation needs.

3. Parametric constraints: Requirements come in many forms, including parametric constraints (i.e. the system delivered must run at speed x) as well as general guidelines (e.g. the system's interface should be user friendly) and domain tasks and processes. Parametric constraints allow a static evaluation to narrow the field of available components.

An example security requirements list:

RequirementID	Requirement Description	Type	Priority
10	Sensitive data is not logged in clear text by the application.	Implementation	Must
20	Database connections, passwords, keys, or other secrets are not stored in plain text.	Business	Must
30	Encryption keys must be secured.	Business	Must
40	Privileged and super-user accounts (Administrator, root, etc.) must not be used for non-administrator activities. A secure mechanism to escalate privileges (e.g., via User Account Control or via sudo) with a standard account is acceptable to meet this requirement. Network services must run under accounts assigned the minimum necessary privileges.	Functional	Should
50	Sensitive data is not stored in persistent cookies.	Business	Wont
60	Sensitive data is transmitted with the HTML	Implementation	Should

	POST protocol. So GET is NOT used for sensitive data.		
70	User ID must be unique. Passwords must be stored in irreversible encrypted form, and the password file cannot be viewed in unencrypted form. A password must not be displayed on the data entry/display device. Passwords must be at least eight characters long. Passwords must be composed of at least three of the following: English uppercase letters, English lowercase letters, numeric characters, and special characters. Password lifetime will not exceed 60 days Users cannot use the previous six passwords. The system will give the user a choice of alternative passwords from which to choose. Passwords must be changed by the user after initial logon.	Business	Must

For this book we started collecting security and privacy requirements, since our experience shows that all good (security) architectures and designs have similar (if not exact) the same requirements. Within the appendix of this document a link to a reusable list of security and privacy requirements on GitHub for reuse. We encourage everyone to share created requirements. See the Appendix on how you can collaborate and make the next version of this reference architecture with us.

Security Principles

Name Principle	Address Privacy&Security
Statement	Address Privacy & Security
Rationale	Information is power and this is certainly true in the context of technology-enabled global development interventions. How information is collected, stored, analysed, shared, and used has serious implications for both the populations about whom data are being transmitted, and the organizations transmitting the data.
Implications	• Assess and mitigate risks to the security of users and their data. • Consider the context and needs for privacy of personally identifiable information when designing solutions and mitigate accordingly. • Ensure equity and fairness in co-creation, and protect the best interests of the end end-users.

Name Principle	Always consider the users
Statement	Always consider the users
Rationale	The security of a software system is linked to what its users do with it. It is therefore important that all security-related mechanisms are designed in a manner that makes it easy for users to deploy, configure, use, and update the system securely. Security is not a feature that can simply be added to a

software system, but rather a property emerging from how the system was built and is operated. The way each user interacts with software is dictated not only by the design and implementation decisions of its creators but also by the cognitive abilities and cultural background of its users.

Failing to address this design principle can lead to a various problems, e.g.:

Implications

- When designers don't "remember the user" in their software design, inadvertent disclosures by the user may take place. If it is difficult to understand the authorization model, or difficult to understand the configuration for visibility of data, then the user's data are likely to be unintentionally disclosed.
- Designers sometimes fail to account for the fact that authenticated and properly authorized users can also be attackers! This design error is a failure to distrust the user, resulting in authorized users having opportunities to misuse the system.
- When security is too hard to set up for a large population of the system's users, it will never be configured, or it will not be configured properly.

Name Principle	**Asset protection and resilience**
Statement	Asset protection and resilience
Rationale	Consumer data, and the assets storing or processing it, should be protected against physical tampering, loss, damage or seizure.
Implications	If this principle is not implemented, inappropriately data (e.g. user or consumer) could be compromised which may result in legal and regulatory sanction, or reputation damage.

Name Principle	Assume that external systems are insecure
Statement	Assume that external systems are insecure.
Rationale	The term information domain arises from the practice of partitioning information resources according to access control, need, and levels of protection required. Organizations implement specific measures to enforce this partitioning and to provide for the flow of authorized information between information domains. The boundary of an information domain represents the security perimeter for that domain. An external domain is one that is not under your control. In general, all external systems should be considered insecure.
Implications	• Take proactive security measurements to protect secure data crossing information boundaries. • Design secure information exchange interfaces (api's). • Make agreements with parties involved.

Name Principle	Audit information provision to consumers
Statement	Audit information provision to consumers
Rationale	Consumers should be provided with the audit records they need to monitor access to their service and the data held within it. If this principle is not implemented, consumers will not be able to detect and respond to inappropriate or malicious use of their service or data within reasonable time-scales. In most countries this is a legal requirement from privacy point

of view.

Implications	• Secure audit mechanism needed. • Requirements needed for audit data retention, storing, archiving.

Name Principle	**Authenticate users and processes**
Statement	Authenticate users and processes to ensure appropriate access control decisions both within and across domains.
Rationale	Authentication is the process where a system establishes the validity of a transmission, message, or a means of verifying the eligibility of an individual, process, or machine to carry out a desired action, thereby ensuring that security is not compromised by an untrusted source. It is essential that adequate authentication be achieved in order to implement security policies and achieve security goals.
Implications	Authentication service needed for users and application processes.

Name Principle	**Authorize after you authenticate**
Statement	Authorize after you authenticate.
Rationale	Authorization should be conducted as an explicit check, and as necessary even after an initial authentication has been completed. Authorization depends not only on the privileges associated with an authenticated user, but also on the context of the request. The time of the request and the

	location of the requesting user may both need to be taken into account.
Implications	For particularly sensitive operations, authorization may need to invoke authentication (again). Although authorization begins only after authentication has occurred, this requirement is not circular. Authentication is not binary—users may be required to present minimal (such as a password) or more substantial (e.g. biometric or token-based) evidence of their identity, and authentication in most systems is not continuous—a user may authenticate, but walk away from the device or hand it to someone else.

Name Principle	**Avoid security by obscurity**
Statement	Security measurements should be open and transparent.
Rationale	• Assume attackers will have source code (also for closed source software). • Assume attackers will have complete design and network topologies. • Open security design promotes cycle of improvement faster. • Assume sensitive information regarding security measurements are leaked or sold.
Implications	• Do not document secrets and configuration policies (settings) in security designs. • Never store secrets (e.g. passwords) on systems. • Involve internal and external SME to evaluate the strength and weakness of a security design. (design review). • Security should always be tested by experts (open or not).

- Periodically pentest the security implementation, use different companies instead of always the same.

Name Principle	Check the return value of functions
Statement	Check the return value of all non-void functions, and check the validity of all function parameters. The return value of non-void functions must be checked by each calling function, and the validity of parameters must be checked inside each function.
Rationale	This is possibly the most frequently violated principle.In the strictest interpretation, this rule means that even the return value of printf statements and file close statements must be checked. A case can be made, though, that if the response to an error would rightfully be no different than the response to success, there is no point in checking a return value. This is often the case with calls to printf and close. In cases like these, it can be acceptable to explicitly cast the function return value to (void) -- thereby indicating that the programmer explicitly and not accidentally decides to ignore a return value. The rule is then only violated if the cast is missing. In more dubious cases, a comment should be present to explain why a return value is irrelevant. In most cases, though, the return value of a function should not be ignored, especially if error return values must be propagated up the function call chain. Standard libraries famously violate this rule with potentially grave consequences. See, for instance, what happens if you accidentally execute strlen(0), or strcat(s1, s2, -1) with the standard C string library. For this reason, most coding guidelines for safety critical software also forbid the use of all ansi standard headers like string.h, stdlib.h, stdio.h etc. If the function are needed, they should be written separately, and made compliant with safety critical use. The enforcement of this principle make sure that exceptions are always explicitly justified (and justifiable), with mechanical checkers flagging violations. Often, it will be easier to comply with the rule than to explain why non-compliance is acceptable.
Implications	• Extra testing and programming effort:Function parameters should normal be verified for validity before being used. This rule

especially applies to pointers: before dereferencing a pointer that is passed as a parameter the pointer must be checked for null.

- Consider automating security testing on software (static and dynamic tests)

Name Principle	**Clearly delineate the physical and logical security boundaries**
Statement	Clearly delineate the physical and logical security boundaries governed by associated security policies.
Rationale	Information technology exists in physical and logical locations, and boundaries exist between these locations. An understanding of what is to be protected from external factors can help ensure adequate protective measures are applied where they will be most effective. Sometimes a boundary is defined by people, information, and information technology associated with one physical location.
Implications	Create a security architecture or design.

Name Principle	**Compartmentalise**
Statement	Sub-systems will be partitioned logically and isolated using physical devices and/or security controls.
Rationale	In accordance with the minimise attack surface and Defence in Depth principles, this compartmentalise principle keeps a sub-system, or logically grouped set of sub-systems, relatively self-contained such that compromise

of one will not imply the compromise of another.

Implications

- Use defence in depth security principles in the security architecture.
- Sourcing of (sub)systems is easily possible when this principles is implemented correctly.
- Eliminate or minimize dependencies between subsystems. This can result in using other (generic) security services like a separate identification or authentication service.

Name Principle	**Compile with all warnings enabled**
Statement	Compile with all warnings enabled, in pedantic mode, and use one or more modern static source code analyzers. All code must be compiled, from the first day of development, with all compiler warnings enabled at the compiler's most pedantic setting. All code must compile with these setting without warnings. All code must be checked on each build with at least one, but preferably more than one, state-of-the-art static source code analyzer and should pass the analyses with zero warnings.
Rationale	There are several very effective static source code analyzers on the market today, and quite a few freeware tools as well. There is no excuse for any serious software development effort not to make use of this technology. It should be considered routine practice, especially for critical software development. The rule of zero warnings applies even in cases where the compiler or the static analyzer gives an erroneous warning: if the compiler or the static analyzer gets confused, the code causing the confusion should be rewritten so that it becomes more trivially valid. Many have been caught in the assumption that a warning was likely invalid, only to realize much later that the report was in fact valid for less obvious reasons. Static analyzers originally had a bad reputation due to the limited capabilities of early versions (e.g., the early Unix tool lint). The early tools produced mostly invalid messages, but this is not the case for the current generation of

commercial tools. The best static analyzers today are fast, and they produce selective and accurate messages.

Implications Provide awareness trainings of developers continuously.

Name Principle	**Complete mediation**
Statement	Complete mediation
Rationale	Access rights are completely validated every time an access occurs. Systems should rely as little as possible on access decisions retrieved from a cache. Again, file permissions tend to reflect this model: the operating system checks the user requesting access against the file's ACL. The technique is less evident when applied to email, which must pass through separately applied packet filters, virus filters, and spam detectors.
Implications	• Document decisions regarding use of cached data for security services. • Usability aspects should be taken into account with setting cache invalidation timers.

Name Principle	**Computer security is constrained by societal factors**
Statement	Computer Security is Constrained by Societal Factors.
Rationale	The ability of security to support the mission of an organization may be limited by various factors, such as social issues. For example, security and

	workplace privacy can conflict. Commonly, security is implemented on an IT system by identifying users and tracking their actions. However, expectations of privacy vary and can be violated by some security measures. (In some cases, privacy may be mandated by law.)
Implications	• User awareness campaigns should be included in the security processes on regular basis. • IT security measurements are a part of the total security system. Organization processes and policies are of great importance.

Name Principle	**Computer Security Requires a Comprehensive and Integrated Approach**
Statement	Computer Security Requires a Comprehensive and Integrated Approach
Rationale	Providing effective computer security requires a comprehensive approach that considers a variety of areas both within and outside of the computer security field. This comprehensive approach extends throughout the entire information life cycle. To work effectively, security controls often depend upon the proper functioning of other controls. Many such interdependencies exist. If appropriately chosen, managerial, operational,and technical controls can work together synergistically.
Implications	The effectiveness of security controls (also) depends on such factors as system management, legal issues, quality assurance, and internal and management controls. Computer security needs to work with traditional security disciplines including physical and personnel security.

Name	**Computer Security Responsibilities and Accountability Should Be Made**

Principle	Explicit
Statement	Computer Security Responsibilities and Accountability Should Be Made Explicit
Rationale	The responsibility and accountability3 of owners, providers, and users of IT systems and other parties4 concerned with the security of IT systems should be explicit.5 The assignment of responsibilities may be internal to an organization or may extend across organizational boundaries.
Implications	Depending on the size of the organization, the computer security program may be large or small, even a collateral duty of another management official. However, even small organizations can prepare a document that states organization policy and makes explicit computer security responsibilities.

Name Principle	Computer Security Should Be Cost-Effective
Statement	Computer Security Should Be Cost-Effective.
Rationale	The costs and benefits of security should be carefully examined in both monetary and nonmonetary terms to ensure that the cost of controls does not exceed expected benefits. Security should be appropriate and proportionate to the value of and degree of reliance on the IT systems and to the severity, probability, and extent of potential harm. Requirements for security vary, depending upon the particular IT system.
Implications	• Calculated the cost of damage against security measurements. • Take notice of legal boundaries possible and lawsuits possible (for liability) if no adequate security measurements are taken. • Consider using proven generic OSS security services when applicable.

Name Principle	Computer Security should be periodically reassessed
Statement	Computer Security Should Be Periodically reassessed
Rationale	Computers and the environments in which they operate are dynamic. System technology and users, data and information in the systems, risks associated with the system, and security requirements are ever-changing. Many types of changes affect system security: technological developments (whether adopted by the system owner or available for use by others); connection to external networks; a change in the value or use of information; or the emergence of a new threat. In addition, security is never perfect when a system is implemented.
Implications	Implement security audits and pentest with your security control processes.

Name Principle	Computer Security Supports the Mission of the Organization
Statement	Computer Security Supports the Mission of the Organization.
Rationale	The purpose of computer security is to protect an organization's valuable resources, such as information, hardware, and software. Through the selection and application of appropriate safeguards, security helps the organization's mission by protecting its physical and financial resources, reputation, legal position, employees, and other tangible and intangible assets.
Implications	IT Security should like all other IT services enable to business to run their processes. So an enabling service and not a disabler service.

Name Principle	Data in transit protection
Statement	Data in transit protection
Rationale	Consumer data transiting networks should be adequately protected against tampering and eavesdropping via a combination of network protection and encryption.
Implications	If this principle is not implemented, then the integrity or confidentiality of the data may be compromised whilst in transit.

Name Principle	Data is always protected
Statement	Data is protected from unauthorized use and disclosure. In addition to the traditional aspects of data classification, this includes, but is not limited to, protection of per-decisional, sensitive, source selection-sensitive, and proprietary information.
Rationale	Open sharing of information and the release of information via relevant legislation must be balanced against the need to restrict the availability of classified, proprietary, and sensitive information. Existing laws and regulations require the safeguarding of security and the privacy of data, while permitting free and open access.
Implications	Aggregation of data, both classified and not, will create a large target requiring review and de-classification procedures to maintain appropriate control. Access to information based on a need-to-know policy will force regular reviews of the body of information. Security needs must be identified

and developed at the data level, not the application level. Data security safeguards can be put in place to restrict access to "view only", or "never see". Sensitivity labelling of data for access to pre-decisional, decisional, classified, sensitive, or proprietary information must be determined. Security must be designed into data elements from the beginning; it cannot be added later. Systems, data, and technologies must be protected from unauthorized access and manipulation. Headquarters information must be safeguarded against inadvertent or unauthorized alteration, sabotage, disaster, or disclosure.

Name Principle	**Declare data objects at the smallest possible level of scope**
Statement	Declare data objects at the smallest possible level of scope.
Rationale	Basic principle of data-hiding. Clearly if an object is not in scope, its value cannot be referenced or corrupted. Similarly, if an erroneous value of an object has to be diagnosed, the fewer the number of statements where the value could have been assigned; the easier it is to diagnose the problem. The rule discourages the re-use of variables for multiple, incompatible purposes, which can complicate fault diagnosis.
Implications	Data should always be declared at the start of the scope in which it is used: for file scope, the declarations go at the top of the source file (never in a header file); for function scope, the declaration goes at the top of the function body; for block scope, at the start of the block. This means that declarations should not be placed at random places in the code, e.g., that the point of first use. Data objects only used in one file should be declared file *static*.

Name	**Defense in depth**

Principle	
Statement	Defense in depth should be a key architecture and design principle.
Rationale	Multi-layered security controls and practices are better than single defense layer.
Implications	• Do not trust on security measurements from preceding functions. • Prepare for the worst possible scenario. • Implement multiple defence mechanism. • Create a security architecture or design and document the different layers of protection. • If one security service fails, the security system should still be resistant against threads. • Compartmentalize and work with secure boundaries for information flows.

Name Principle	Design and implement audit mechanisms
Statement	Design and implement audit mechanisms to detect unauthorized use and to support incident investigations.
Rationale	Organizations should monitor, record, and periodically review audit logs to identify unauthorized use and to ensure system resources are functioning properly. In some cases, organizations may be required to disclose information obtained through auditing mechanisms to appropriate third parties.

Implications	• Audit logs must be protected against manipulation. (online/offline). • All audit records should have a correct time stamp. • Unified time service is needed for a secure audit service. • Integrity of the audit system must be implemented.

Name Principle	**Design and operate an IT system to limit damage and to be resilient in response.**
Statement	Design and operate an IT system to limit damage and to be resilient in response.
Rationale	Information systems should be resistant to attack, should limit damage, and should recover rapidly when attacks do occur. The principle suggested here recognizes the need for adequate protection technologies at all levels to ensure that any potential cyber attack will be countered effectively.
Implications	• Defence in depth measurement • Compartmentalize IT building blocks.

Name Principle	**Design for secure updates**
Statement	Design for secure updates

Rationale	All updates for a system must be verified. The source of the update must be known and the integrity must be verified. It is easier to upgrade small pieces of a system than huge blobs. Doing so ensures that the security implications of the upgrade are well understood and controlled.
Implications	• Verify the integrity and provenance of upgrade packages. • Make use of code signing and signed manifests to ensure that the system only consumes patches and updates of trusted origin. E.g. use secure hashing (sha).

Name Principle	**Design for security properties changing over time**
Statement	Design for security properties changing over time
Rationale	The migration of previous users (and/or the correct coexistence of the local and remote users) would need to happen in a way that does not compromise security.
Implications	Make security design modular and flexible from the start.

Name Principle	**Design reviews**
Statement	All architectures and designs must be reviewed. Minimal on security aspects and potential risks. Also to determine if all (security and privacy) principles and requirements are followed.

Rationale	Integrating security into the design phase saves money and time. Conduct a risk review with security professionals and threat model the application to identify key risks and to improve product and processes under development. This helps you integrate appropriate countermeasures into the design and architecture of the application. Improving architecture and design is by far the best option (time,cost etc) for dealing with security and privacy.
Implications	Organize or make use of a structured review process to benefit from review. SME (Subject Matter Experts) must be available for doing reviews. Reserve time to improve architectures and designs or to improve code.

Name Principle	**Design security to allow for regular adoption of new technology**
Statement	Design security to allow for regular adoption of new technology, including a secure and logical technology upgrade process.
Rationale	As mission and business processes and the threat environment change, security requirements and technical protection methods must be updated. IT-related risks to the mission/business vary over time and undergo periodic assessment.
Implications	

Name Principle	**Develop and exercise contingency or disaster recovery procedures to ensure appropriate availability**
Statement	Develop and exercise contingency or disaster recovery procedures to ensure appropriate availability

Rationale	Continuity of operations plans or disaster recovery procedures address continuance of an organization's operation in the event of a disaster or prolonged service interruption that affects the organization's mission.
Implications	

Name Principle	**Do not implement unnecessary security mechanisms.**
Statement	Do not implement unnecessary security mechanisms.
Rationale	Every security mechanism should support a security service or set of services, and every security service should support one or more security goals. Extra measures should not be implemented if they do not support a recognized service or security goal. Such mechanisms could add unneeded complexity to the system and are potential sources of additional vulnerabilities.
Implications	Only implement security measurements when needed.

Name Principle	**Don't trust infrastructure**
Statement	Underlaying infrastructure cannot be assumed safe.
Rationale	Vulnerabilities are at hardware,firmwire, virtualization, middleware and application layers. To minimize data leakage risks trusting security of other objects should be prevented.

Implications	Sandbox model /Jericho model needed. Layered defense easily possible

Name Principle	**Don't trust services (from others)**
Statement	Services from others (departments, companies) should never (ever) be trusted.
Rationale	Security design should protect against services use of other layers or applications (also SAAS services). Systems or sub-systems outside the bounds of a receiving component must never be trusted implicitly.
Implications	Every input/output and given by external services must be validated. Authentication, authorization can be needed. Measurements to maintain availability when using services (input or output) requires strict measurements implemented.

Name Principle	**Earn or give, but never assume or trust**
Statement	Earn or give, but never assume or trust
Rationale	Offloading security functions from server to client exposes those functions to a much less trustworthy environment, which is one of the most common causes of security failures predicated on misplaced trust. Designs that place authorization, access control,enforcement of security policy, or embedded sensitive data in client software thinking that it won't be discovered, modified, or exposed by clever users or malicious attackers are inherently weak. Such designs will often lead to compromises.

Implications	• Make sure all data received from an untrusted client are properly validated before processing. • When designing your systems, be sure to consider the context where code will be executed, where data will go, and where data entering your system comes from.

Name Principle	**Economy of mechanism**
Statement	A simple design is easier to test and validate.
Rationale	Keep it simple to avoid risk. More is not always better. This means more components, more processes and more security measurements involved. One factor in evaluating a system's security is its complexity. If the design, implementation, or security mechanisms are highly complex, then the likelihood of security vulnerabilities increases. Simpler means less can go wrong. This well-known principle applies to any aspect of a system, but it deserves emphasis for protection mechanisms for this reason: design and implementation errors that result in unwanted access paths will not be noticed during normal use (since normal use usually does not include attempts to exercise improper access paths).
Implications	Avoid complexity.

Name Principle	**Ensure proper security in the shutdown or disposal of a system**
Statement	Ensure proper security in the shutdown or disposal of a system

Rationale	Although a system may be powered down, critical information still resides on the system and could be retrieved by an unauthorized user or organization. Access to critical information systems must be controlled at all times.
Implications	• At the end of a system's life-cycle, system designers should develop / design procedures to dispose of an information system's assets in a proper and secure fashion. • Procedures must be implemented to ensure system hard drives, volatile memory, and other media are purged to an acceptable level and do not retain residual information.

Name Principle	**Ensure that developers are trained in how to develop secure software.**
Statement	Ensure that developers are trained in how to develop secure software.
Rationale	It is unwise to assume that developers know how to develop secure software. Therefore, ensure that developers are adequately trained in the development of secure software before developing the system. This includes application of engineering disciplines to design, development, configuration control, and integration and testing.
Implications	Training cost (permanent) for all staff involved in maintaining the IT assets of a company.

Name Principle	**Establish a sound security policy as the"foundation" for design.**

Statement	Establish a sound security policy as the “foundation” for design.
Rationale	A security policy is an important document to develop while designing an information system. The security policy begins with the organization’s basic commitment to information security formulated as a general policy statement. The policy is then applied to all aspects of the system design or security solution. The policy identifies security goals (e.g., confidentiality, integrity, availability, accountability, and assurance) the system should support, and these goals guide the procedures, standards and controls used in the IT security architecture design. The policy also should require definition of critical assets, the perceived threat, and security-related roles and responsibilities.
Implications	A security architecture or security design should be based on requirements that are derived from the policies defined or directly of the policies.

Name Principle	**Establish secure defaults**
Statement	Establish secure defaults when system goes in error or exception status, or at default start-up.
Rationale	Secure defaults lower the risk of bad configurations.
Implications	• Security design principles and requirements must be implemented at first release. • Installation of software without safe defaults is not possible. • Secure defaults must be determined and configured. • Secure defaults must be regularly tested

Name Principle	External interface protection
Statement	External interface protection
Rationale	All external or less trusted interfaces of the service should be identified and have appropriate protections to defend against attacks through them. If this principle is not implemented, interfaces could be subverted by attackers in order to gain access to the service or data within it.
Implications	

Name Principle	Fail Safe Defaults
Statement	Fail Safe Defaults
Rationale	A mechanism that, in the event of failure, responds in a way that will cause no harm, or at least a minimum of harm, to other devices or danger to personnel.
Implications	• Stress under load and hard failure situations must be incorporated in the security test suite. • Default system configuration at start-up is secure.

Name Principle	Fail-safe default settings for security and access
Statement	Fail-safe default settings for security and access. So in case of error security should not be compromised.
Rationale	In computing systems, the save default is generally “no access” so that the system must specifically grant access to resources. Most file access permissions work this way, though Windows also provides a “deny” right. Windows access control list (ACL) settings may be inherited, and the “deny” right gives the user an easy way to revoke a right granted through inheritance. However, this also illustrates why “default deny” is easier to understand and implement, since it’s harder to interpret a mixture of “permit” and “deny” rights.
Implications	

Name Principle	Formulate security measures to address multiple overlapping information domains
Statement	Formulate security measures to address multiple overlapping information domains.
Rationale	An information domain is a set of active entities (person, process, or devices) and their data objects. A single information domain may be subject to multiple security policies. A single security policy may span multiple information domains. An efficient and cost effective security capability should be able to enforce multiple security policies to protect multiple information domains without the need to separate (physically or logically) the information and respective information systems processing the data.
Implications	

Name Principle	Governance framework
Statement	A Governance framework is required for service providers of Cloud hosting.
Rationale	The service provider should have a security governance framework that coordinates and directs their overall approach to the management of the service and information within it. If this principle is not implemented, any procedural, personnel, physical and technical controls in place will not remain effective when responding to changes in the service and to threat and technology developments.
Implications	

Name Principle	HTTP header use
Statement	HTTP header information is not relied on to make security decisions.
Rationale	HTTP headers can be manipulated very easily.
Implications	Test if software does not make security decisions based on HTTP headers. Perform e.g. security tests with manipulated headers.

Name	Identify and prevent common errors and vulnerabilities

Principle	
Statement	Identify and prevent common errors and vulnerabilities
Rationale	Many errors reoccur with disturbing regularity - errors such as buffer overflows, race conditions, format string errors, failing to check input for validity, and programs being given excessive privileges. Learning from the past will improve future results.
Implications	Use OWASP top 10 checklist Use proven security test tools that are regular updated.

Name Principle	Identify potential trade-offs
Statement	Identify potential trade-offs between reducing risk and increased costs and decrease in other aspects of operational effectiveness.
Rationale	To meet stated security requirements, a systems designer, architect, or security practitioner will need to identify and address all competing operational needs. It may be necessary to modify or adjust (i.e., trade-off) security goals due to other operational requirements. In modifying or adjusting security goals, an acceptance of greater risk and cost may be inevitable.
Implications	Document all relevant design decisions within a maintained security architecture or design document.

Name	Identity and authentication

Principle	
Statement	Identity and authentication
Rationale	Access to all service interfaces (for consumers and providers) should be constrained to authenticated and authorised individuals. If this principle is not implemented, unauthorised changes to a consumer's service, theft or modification of data, or denial of service may occur.
Implications	

Name Principle	Implement layered security (Ensure no single point of vulnerability).
Statement	Implement layered security (Ensure no single point of vulnerability).
Rationale	Security designs should consider a layered approach to address or protect against a specific threat or to reduce vulnerability. For example, the use of a packet-filtering router in conjunction with an application gateway and an intrusion detection system combine to increase the work-factor an attacker must expend to successfully attack the system.
Implications	

Name Principle	Implement least privilege
Statement	Implement least privilege.

Rationale	The concept of limiting access, or "least privilege," is simply to provide no more authorizations than necessary to perform required functions. This is perhaps most often applied in the administration of the system. Its goal is to reduce risk by limiting the number of people with access to critical system security controls; i.e., controlling who is allowed to enable or disable system security features or change the privileges of users or programs. Best practice suggests it is better to have several administrators with limited access to security resources rather than one person with "super user" permissions.
Implications	

Name Principle	**Implement tailored system security measures to meet organizational security goals.**
Statement	Implement tailored system security measures to meet organizational security goals.
Rationale	In general, IT security measures are tailored according to an organization's unique needs. While numerous factors, such as the overriding mission requirements, and guidance, are to be considered, the fundamental issue is the protection of the mission or business from IT security related, negative impacts.
Implications	

Name Principle	**Isolate public access systems from mission critical resources**
Statement	Isolate public access systems from mission critical resources (e.g., data,

	processes, etc.).
Rationale	While the trend toward shared infrastructure has considerable merit in many cases, it is not universally applicable. In cases where the sensitivity or criticality of the information is high, organizations may want to limit the number of systems on which that data is stored and isolate them, either physically or logically. Physical isolation may include ensuring that no physical connection exists between an organization's public access information resources and an organization's critical information. When implementing logical isolation solutions, layers of security services and mechanisms should be established between public systems and secure systems responsible for protecting mission critical resources.
Implications	Isolation measurements must be tested regularly. An audit report from a third party is required (in case of cloud sourcing).

Name Principle	**Least common mechanism**
Statement	Least common mechanism
Rationale	Users should not share system mechanisms except when absolutely necessary, because shared mechanisms may provide unintended communication paths or means of interference.
Implications	

Name Principle	**Least privilege**

Statement	Least privilege
Rationale	Every program and user should operate while invoking as few privileges as possible. This is the rationale behind Unix “sudo” and Windows User Account Control, both of which allow a user to apply administrative rights temporarily to perform a privileged task.
Implications	This principle has impact on the system, software components, but also on procedures used.

Name Principle	**Limit the use of pointers**
Statement	Limit the use of pointers. Use no more than N levels of dereferencing (star operators) per expression. A strict value for N=1, but in some cases using N=2 can be justified. Pointer dereference operations may not be hidden in macro definitions or inside typedef declarations. The use of function pointers should be restricted to simple cases.
Rationale	Pointers are easily misused, even by experienced programmers. They can make it hard to follow or analyze the flow of data in a program, especially by tool-based static analyzers. Function pointers, similarly, can seriously restrict the types of checks that can be performed by static analyzers and should only be used if there is a strong justification for their use, and ideally alternate means are provided to assist tool-based checkers determine flow of control and function call hierarchies. For instance, if function pointers are used, it can become impossible for a tool to prove absence of recursion, so alternate guarantees would have to be provided to make up for this loss in analytical capabilities.
Implications	It should be possible for a static analyzer to determine in all cases which function is being called, if the call is made through a function pointer. It may be acceptable to allow cases where the number of possible functions that may be called is larger than one, provided it does not affect the precision of

the code analysis itself. This means that it can depend on the capabilities of a specific static analyzer what liberties can be taken with the use of function pointers. Additionally, though, it is wise to keep function pointer use to a minimum, and to restrict to simple cases, to make sure that also humans can determine accurately and with modest effort which functions may be evoked.

Name Principle	**Limit the use of the preprocessor to file inclusion and simple macros**
Statement	Limit the use of the preprocessor to file inclusion and simple macros. The use of the preprocessor must be limited to the inclusion of header files and simple macro definitions. Token pasting, variable argument lists (ellipses), and recursive macro calls are not permitted. All macros must expand into complete syntactic units. The use of conditional compilation directives should be restricted to the prevention of duplicate file inclusion in header files.
Rationale	The C preprocessor is a powerful obfuscation tool that can destroy code clarity and befuddle many text based checkers. The effect of constructs in unrestricted preprocessor code can be extremely hard to decipher, even with a formal language definition in hand. In a new implementation of the C preprocessor, developers often have to resort to using earlier implementations as the referee for interpreting complex defining language in the C standard. The rationale for the caution against conditional compilation is equally important. Note that with just ten conditional compilation directives, there could be up to 2^10 (i.e., 1024) possible versions of the code, each of which would have to be tested -- causing a significant increase in the required test effort.
Implications	Macros should only appear in header files, never in the source code itself. The #undef directive should not be used. Macros should never hide declarations, and they should not hide pointer dereference operations from the code. Macros should also never be used to redefine the language. The restriction of macro definitions to the definition of complete syntactic units means that *all* macro bodies must be enclosed in either round or curly

braces. **Compiler directives** There should not be more #ifdef directives in a code base than there are headerfiles. Each use of compilation directives (other than the duplicate file inclusion prevention use) should be flagged by a tool-based checker and justified with a comment in the code.

Name Principle	**Logging secrets**
Statement	Private data (for example, passwords) is not logged.
Rationale	Protecting secure logs is expensive.
Implications	A clear message level must be built in to notify exactly what the cause of error is. Reduced risk profile on system logs.

Name Principle	**Minimize secrets**
Statement	Minimize secrets
Rationale	Secrets should be few and changeable, but they should also maximize entropy, and thus increase the attacker's work factor. The simple principle is also true by itself, since each secret increases a system's administrative burden.
Implications	

Name Principle	Minimize the system elements to be trusted.
Statement	Minimize the system elements to be trusted.
Rationale	Security measures include people, operations, and technology. Where technology is used, hardware, firmware, and software should be designed and implemented so that a minimum number of system elements need to be trusted in order to maintain protection.
Implications	

Name Principle	Open design
Statement	Open design. The security of physical products, machines and systems should not depend on secrecy of the design and implementation.
Rationale	Baran (1964) argued persuasively in an unclassified RAND report that secure systems, including cryptographic systems, should have unclassified designs. This reflects recommendations by Kerckhoffs (1883) as well as Shannon's maxim: "The enemy knows the system" (Shannon, 1948). Even the NSA, which resisted open crypto designs for decades, now uses the Advanced Encryption Standard to encrypt classified information.
Implications	

Name	Operational security

Principle	
Statement	Operational security
Rationale	The service provider should have processes and procedures in place to ensure the operational security of the service. processes and procedures in place to ensure the operational security of the service. If this principle is not implemented, the service can't be operated and managed securely in order to impede, detect or prevent attacks against it.
Implications	

Name Principle	Personnel security
Statement	Personnel security
Rationale	Service provider staff should be subject to personnel security screening and security education for their role. If this principle is not implemented, the likelihood of accidental or malicious compromise of consumer data by service provider personnel is increased.
Implications	

Name Principle	Protect information while being processed, in transit, and in storage.
Statement	Protect information while being processed, in transit, and in storage.

Rationale	The risk of unauthorized modification or destruction of data, disclosure of information, and denial of access to data while in transit should be considered along with the risks associated with data that is in storage or being processed. Therefore, system engineers, architects, and IT specialists should implement security measures to preserve, as needed, the integrity, confidentiality, and availability of data, including application software, while the information is being processed, in transit, and in storage.
Implications	

Name Principle	**Provide assurance that the system is, and continues to be, resilient in the face of expected threats.**
Statement	Provide assurance that the system is, and continues to be, resilient in the face of expected threats.
Rationale	Assurance is the grounds for confidence that a system meets its security expectations. These expectations can typically be summarized as providing sufficient resistance to both direct penetration and attempts to circumvent security controls. Good understanding of the threat environment, evaluation of requirement sets, hardware and software engineering disciplines, and product and system evaluations are primary measures used to achieve assurance. Additionally, the documentation of the specific and evolving threats is important in making timely adjustments in applied security and strategically supporting incremental security enhancements.
Implications	Security testing must be planned and performed on regular basis.

Name	**Psychological acceptability**

Principle	
Statement	Psychological acceptability
Rationale	This principle essentially requires the policy interface to reflect the user's mental model of protection, and notes that users won't specify protections correctly if the specification style doesn't make sense to them.
Implications	

Name Principle	**Reduce risk to an acceptable level.**
Statement	Reduce risk to an acceptable level.
Rationale	Risk is defined as the combination of (1) the likelihood that a particular threat source will exercise (intentionally exploit or unintentionally trigger) a particular information system vulnerability and (2) the resulting adverse impact on organizational operations, organizational assets, or individuals should this occur.
Implications	

Name Principle	**Risk Based Approach to Security**
Statement	Ensure that risks to confidentiality, integrity, and availability of information and technology systems are treated in a consistent and effective manner.

Risk is the chance of something happening that will have an impact on company objectives and risk assessment is the overall process of risk identification, analysis, evaluation, and mitigation.

Rationale

- Taking a risk based approach allows for the: better identification of threats to our projects and initiatives,
- more effective allocation and use of resources to manage those risks, and
- improved stakeholder confidence and trust as we better manage information and business risk.

The level and cost of information security controls to manage confidentiality, integrity, and availability risk must be appropriate and proportionate to the value of the information assets and the potential severity, probability, and extent of harm. Risks must identified so we are aware of what risks can occur, what existing controls are in place, the consequence and likelihood of the risk occurring, and a determination is made about how to treat those risks.

Implications

- Options for addressing information risk should be reviewed so that informed and documented decisions are made about the treatment of risk. Risk treatment involves choosing one or more options, which typically include: Accepting risk (by an appropriate team member signing off that he/she has accepted the risk and no further action is required)
- Avoiding risk (by an appropriate team member deciding not to pursue a particular initiative)
- Transferring risk (by an appropriate team member to an external entity such as insurance)
- Mitigating risk (by an appropriate team member by applying appropriate information security measures, e.g., access controls, network monitoring and incident management)

Name Principle	Secure use of the service by the consumer
Statement	Secure use of the service by the consumer
Rationale	Consumers have certain responsibilities when using a cloud service in order for this use to remain secure, and for their data to be adequately protected. If this principle is not implemented, the security of cloud services and the data held within them can be undermined by poor use of the service by consumers.
Implications	

Name Principle	Security by Design
Statement	Controls for the protection of confidentiality, integrity, and availability should be designed into all aspects of solutions from initiation, not as an afterthought. Security should also be designed into the business processes within which an IT system will be used.
Rationale	The implementation of protections for confidentiality, availability and integrity within information and systems at the end of a project is more expensive than including the security protections within the initial design of the project. Controls implemented at the end of a project are often less efficient and less integrated than those integrated within the core of the project.
Implications	• Security is designed in as an integrated part of the system architecture, not added as an afterthought. • Security mechanisms must span all tiers of the architecture, and must be scalable.

- All solutions, custom or commercial, must be tested for security.
- Possible areas of control which could be addressed and integrated include (but are not limited to): asset management and information classification; physical security; segregation of duties, protections against malicious code; backup; exchange of information; logging and monitoring; user access management; technical vulnerability management; compliance with legal requirements; and, information systems audit considerations.

Name Principle	**Sensitive Data**
Statement	Secrets are not stored in code.
Rationale	Storing secrets involves risk at all times.
Implications	Software code must be scanned on secrets (e.g. configuration details, passwords)

Name Principle	**Sensitive data must be identified**
Statement	Sensitive data must be identified and it should be defined how the data is handled.
Rationale	Data sets do not exist only at rest, but in transit between components within a single system and between organizations. As data sets transit between systems, they may cross multiple trust boundaries. Identifying these boundaries and rectifying them with data protection policies is an essential design activity. Trust is just as tricky as data sensitivity, and the notion of

	trust enclaves is likely to dominate security conversations in the next decade.
Implications	Policy requirements and data sensitivity can change over time as the business climate evolves, as regulatory regimes change, as systems become increasingly interconnected, and as new data sources are incorporated into a system. Regularly revisiting and revising data protection policies and their design implications is essential.

Name Principle	**Separation between consumers**
Statement	Separation between consumers
Rationale	Separation should exist between different consumers of the service to prevent one malicious or compromised consumer from affecting the service or data of another.If this principle is not implemented, service providers cannot prevent a consumer of the service affecting the confidentiality or integrity of another consumer’s data or service.
Implications	Sharing services between customers by Cloud Service Providers (CSP's) requires strict separation within the security model.

Name Principle	**Separation of privilege**
Statement	Separation of privilege
Rationale	A protection mechanism is more flexible if it requires two separate keys to unlock it, allowing for two-person control and similar techniques to prevent

unilateral action by a subverted individual. The classic examples include dual keys for safety deposit boxes and the two-person control applied to nuclear weapons and Top Secret crypto materials. A protection mechanism is more flexible if it requires two separate keys to unlock it, allowing for two-person control and similar techniques to prevent unilateral action by a subverted individual. The classic examples include dual keys for safety deposit boxes and the two-person control applied to nuclear weapons and Top Secret crypto materials. Separation of privilege gives better data protection for internal fraud or internal hacks.

Implications

- Security procedures are needed.
- Business Continuity and Disaster Recovery involve more effort.
- Reaction time in case of an incident can be reduced.

Name Principle	**Session lifetime**

Statement Session lifetime is limited. Also for cookies.

Rationale Security System performance

Implications All transactions must be completed within max session time.

Name Principle	**Strive for operational ease of use.**

Statement Strive for operational ease of use.

Rationale	The more difficult it is to maintain and operate a security control, the less effective that control is likely to be. Therefore, security controls should be designed to be consistent with the concept of operations and with ease-of-use as an important consideration.
Implications	

Name Principle	**Strive for simplicity**
Statement	Strive for simplicity
Rationale	The more complex the mechanism, the more likely it may possess exploitable flaws. Simple mechanisms tend to have fewer exploitable flaws and require less maintenance. Further, because configuration management issues are simplified, updating or replacing a simple mechanism becomes a less intensive process.
Implications	

Name Principle	**Supply chain security**
Statement	Supply chain security
Rationale	The service provider should ensure that its supply chain satisfactorily supports all of the security principles that the service claims to implement. If this principle is not implemented, it is possible that supply chain compromise can undermine the security of the service and affect the

implementation of other security principles.

Implications

Name Principle	Systems Owners Have Security Responsibilities Outside Their Own Organizations
Statement	Systems Owners Have Security Responsibilities Outside Their Own Organizations
Rationale	If a system has external users, its owners have a responsibility to share appropriate knowledge about the existence and general extent of security measures so that other users can be confident that the system is adequately secure. This does not imply that all systems must meet any minimum level of security, but does imply that system owners should inform their clients or users about the nature of the security.
Implications	Managers "should act in a timely, coordinated manner to prevent and to respond to breaches of security" to help prevent damage to others.2 However, taking such action should not jeopardize the security of systems.

Name Principle	Treat security as an integral part of the overall system design.
Statement	Treat security as an integral part of the overall system design.
Rationale	Security must be considered in information system design. Experience has shown it to be both difficult and costly to implement security measures properly and successfully after a system has been developed, so it should be integrated fully into the system life-cycle process. This includes establishing

security policies, understanding the resulting security requirements, participating in the evaluation of security products, and finally in the engineering, design, implementation, and disposal of the system.

Implications

Name Principle	**Use an authentication mechanism that cannot be bypassed**
Statement	Use a authentication mechanism that cannot be bypassed or tampered with.
Rationale	The ability to bypass an authentication mechanism can result in an unauthorized entity having access to a system or service that it shouldn't.
Implications	• It's preferable to have a single method, component, or system responsible for authenticating users. Such a single mechanism can serve as a logical "choke point" that cannot be bypassed. • Much as in code reuse, once a single mechanism has been determined to be correct, it makes sense to leverage it for all authentication.

Name Principle	**Use only Secure Protocols**
Statement	Only inherently secure protocols should be used. The protocol should not encapsulate another insecure protocol (IPSec / VPN etc.) The protocol should be capable of authenticating itself

Rationale	Insecure protocols introduce security risks than can be easily avoided.
Implications	Insecure Protocols (http for example) Only used where interaction with non-trusted environment essential. Protocol must be validated against application

Name Principle	**Use standard solutions**
Statement	Existing security controls should be given preference over custom solutions
Rationale	Secure software is hard. The largest, most experienced and deep pocketed software developers in the world, both commercial and open source, are constantly patching security vulnerabilities in software that has been in the wild and hardened over many years. It is arguably implausible for developers of a particular system to invent and deliver a security solution that is as good as or better than an off-the-shelf solution. Add to that the need to fully and clearly document how the custom security solution works for maintainers of the software and new developers to comprehend, maintain and extend the solution and the cost of training up those resources.
Implications	

Name Principle	**Use unique identities to ensure accountability**
Statement	Use unique identities to ensure accountability

Rationale	An identity may represent an actual user or a process with its own identity, e.g., a program making a remote access. Unique identities are a required element in order to be able to: • Maintain accountability and traceability of a user or process • Assign specific rights to an individual user or process • Provide for non-repudiation • Enforce access control decisions • Establish the identity of a peer in a secure communications path • Prevent unauthorized users from masquerading as an authorized user.
Implications	

Name Principle	**Where possible, base security on open standards for portability and interoperability.**
Statement	Where possible, base security on open standards for portability and interoperability.
Rationale	For security capabilities to be effective security program designers should make every effort to incorporate interoperability and portability into all security measures, including hardware and software, and implementation practices. In practice an open interface in OSS software (good documented) can be a good alternative to an open standard that lacks solid reference implementations and gives room to different ways of implementing external behaviour.
Implications	• No all Commercial-off-the-shelf (*COTS*) software is usable.

- OSS solutions should provide open interfaces.

Privacy Principles

Name Principle	Access to Personal data
Statement	The organization provides individuals with access to their personal information for review or update.
Rationale	Comply with global or local regulations or legal constrains.
Implications	• Confirmation of individual's identity before access is given to personal information. • Personal information presented in understandable format. • Access provided in reasonable time frame and at a reasonable cost. • Statement of disagreement; the reason for denial should be explained to individuals in writing.

Name Principle	Collection Limitation Principle
Statement	There should be limits to the collection of personal data and any such data should be obtained by lawful and fair means and, where appropriate, with the knowledge or consent of the data subject. (Source:OECDprivacy.org, by Ben Gerber)
Rationale	When collecting many personal data records this will have a significant impact on: • Risks

- Costs

Collecting personal data means end-users trust you (you do no evil). The more data you collect the harder it will be to protect the data for other forms of usages in future.

Implications

- Make an architecture or design that is clear on what data objects are collected for what business process. Do not collect data with purpose of data mining based on vague use cases.
- Make a data design for all data that is collected.

Name Principle	**Collection of personal data**
Statement	Personal information is only collected for the purposes identified in a notice presented to the users.
Rationale	Legal regulation (local, global)

Implications

- document and describe types of information collected and methods of collection
- collection of information by fair and lawful means, including collection from third parties
- inform individuals if information is developed or additional information is acquired

Name Principle	**Defensive data collection**

Statement	Limited data collected from users only for functionality needed.
Rationale	Only collect data what is needed for performing functionality. Limiting data collection prevents risks on data leakage.
Implications	De-identify where and when possible to reduce risk of privacy data concerns. Data must deleted when no longer necessary.

Name Principle	**Design reviews**
Statement	All architectures and designs must be reviewed. Minimal on security aspects and potential risks. Also to determine if all (security and privacy) principles and requirements are followed.
Rationale	Integrating security into the design phase saves money and time. Conduct a risk review with security professionals and threat model the application to identify key risks and to improve product and processes under development. This helps you integrate appropriate countermeasures into the design and architecture of the application. Improving architecture and design is by far the best option (time,cost etc) for dealing with security and privacy.
Implications	Organize or make use of a structured review process to benefit from review. SME (Subject Matter Experts) must be available for doing reviews. Reserve time to improve architectures and designs or to improve code.

Name Principle	**Disclosure to third parties**

Statement	Personal information is disclosed to third parties only for the identified purposes and with implicit or explicit consent of the individual.
Rationale	• Communication with third parties should be made known to the individual • Information should only be disclosed to third parties that have equivalent agreements to protect personal • Information individuals should be aware of any new uses/purposes for the information the organization should take remedial action in response to misuse of personal information by a third party
Implications	Make sure end-users can read , understand and agree with your privacy terms.

Name Principle	**Don't trust infrastructure**
Statement	Underlaying infrastructure cannot be assumed safe.
Rationale	Vulnerabilities are at hardware,firmwire, virtualization, middleware and application layers. To minimize data leakage risks trusting security of other objects should be prevented.
Implications	Sandbox model /Jericho model needed. Layered defense easily possible

Name	**Don't trust services (from others)**

Principle	
Statement	Services from others (departments, companies) should never (ever) be trusted.
Rationale	Security design should protect against services use of other layers or applications (also SAAS services). Systems or sub-systems outside the bounds of a receiving component must never be trusted implicitly.
Implications	Every input/output and given by external services must be validated. Authentication, authorization can be needed. Measurements to maintain availability when using services (input or output) requires strict measurements implemented.

Name Principle	Individual Participation Principle
Statement	An individual should have the right to get clear insight on data collected that relates to him.
Rationale	Users should be informed on what is collected on request. In some countries this is a legal requirement for all companies collecting personal data.
Implications	• User request should be handled within a reasonable time • At a minimal cost • Users should be informed on how data is protected, deleted and what data is shared with other companies.

Name Principle	Management Responsibility
Statement	The organization defines, documents, communicates and assigns accountability for its privacy policies and procedures.
Rationale	Management is responsible for organising processes needed to be compliant for privacy regulations and handling personal data within the company.
Implications	• privacy policies define and document all ten GAPP • review and approval of changes to privacy policies conducted by management • risk assessment process in place to establish a risk baseline and regularly identify new or changing risks to personal data • infrastructure and systems management takes into consideration impacts on personal privacy • privacy awareness training

Name Principle	Monitoring and enforcement
Statement	The organization monitors compliance with its privacy policies and procedures. It also has procedures in place to address privacy-related complaints and disputes.
Rationale	
Implications	• individuals should be informed on how to contact the organization with inquiries, complaints and disputes

- formal process in place for inquires, complaints or disputes
- each complaint is addressed and the resolution is documented for the individual
- compliance with privacy policies, procedures, commitments and legislation is reviewed, documented and reported to management

Name Principle	Purpose Specification Principle
Statement	The purposes for which personal data are collected should be specified not later than at the time of data collection and the subsequent use limited to the fulfilment of those purposes or such others as are not incompatible with those purposes and as are specified on each occasion of change of purpose. (source: http://oecdprivacy.org/)
Rationale	
Implications	The purpose of personal data collection must be clearly defined in an architecture or design. This involves business, functional and IT designs.

Name Principle	Security for privacy
Statement	Personal information is protected against both physical and logical unauthorized access.
Rationale	• privacy policies must address the security of personal

information

- information security programs must include administrative, technical and physical safeguards
- logical access controls in place
- restrictions on physical access
- environmental safeguards
- personal information protected when being transmitted (e.g. mail, internet, public or other non-secure networks)
- security safeguards should be tested for effectiveness at least once annually

Implications

Name Principle	**Security Safeguards**
Statement	Personal data should be protected by reasonable security safeguards against such risks as loss or unauthorised access, destruction, use, modification or disclosure of data.
Rationale	Personal data is valuable.
Implications	Security must be in place. Security control system must be operational. (prevent,detect, react etc)

Name Principle	Use Limitation Principle
Statement	Personal data should not be disclosed, made available or otherwise used for purposes other than those specified in accordance with a) with the consent of the data subject; or b) by the authority of law. (source: http://oecdprivacy.org/)
Rationale	Using personal data for other means than collected introduces extra risks and complexity for your security and privacy operations. Most of the time other use of data is not securely designed.
Implications	

Using Open Source for security and privacy protection

Introduction

To increase and improve security and protect our privacy open source solutions are more and more seen as a very good solution. Within more and more companies worldwide we notice a trends towards adopting open source solutions for security and privacy protection. Governments worldwide cannot depend and trust on closed source software for their security infrastructure anymore. Gartner predicts that by 2016 99% of Global 2000 enterprises will use open source in mission-critical software. So open source solutions for controlling security and privacy are slowly but steady becoming the new de facto standard. As many security experts already known: Transparency and openness increase security protection levels. However there is still a lot of resistance against using open source for business use, especially when it comes down to security and privacy functionality.

This chapter covers facts and demystifies fads regarding open source security and privacy products. When discussing the use of open source products for security and privacy services two important question appear:

1. Why should open source be used for security and privacy functionality?
2. How can the quality of open source products for security and privacy be determined and judged?

OSS quality is a very popular field for PhD students and analyst companies. However we think that also technical experience of practical business use along with deep technical knowledge is required in order to give a good advice for a company.

Of course we have an opinion regarding using open source security and privacy products for serious business use. However opinions are to be discussed and challenged. Always. Within the technical software field sometimes we tend to see things as hard facts. For examples bad written code. Many measurements exist to measure the quality of software code. However does this means that the product is totally useless? When it comes down to software code, all software contains bugs and has more or less quality issues. If you ask an auditor to look at software code, he will write an audit report with findings and recommendations. Always. If you are hungry and go to McDonald they recommend a very tasty bad solution that works temporary. In the end with every problem you face try to find out the real interest of your trusted advisor. Is he biased? Prepossessed to get a certain

result? Always try to get a real in depended security or privacy advisor when it comes down to questions that relate to your vital business risks. Always challenge the advice! When it comes down to business related questions real facts are hard. Advice is always biased. However be warned for fads! Especially within the field of open source software for regular business use. For decades many vendors have created fads regarding open source. Since this message is repeated over and over again sometimes we are weak and store these fads as facts.

Some famous fads regarding open source the use for business use:

- Open Source software is created by communist to destroy our world.
- Open Source software is made by hobbyist.
- Open Source software is made by hackers and hackers are bad. Especially when it comes down to security and privacy.
- Open Source software is never maintained.
- Open Source software is free, so it can not have any value.
- Quality of Open Source software is dramatic. Do does hackers known how to spell quality at all?
- Using Open Source makes you depended of the good will of hackers.
- Using Open Source for security or privacy protection gives unacceptable high risk, since the whole world can hack me now instantaneously.
- Using Open Source is an extra thread for my security or privacy.

Unfortunately, the list is endless long. Fighting fads is hard. Fads are most of the time a perception based on incorrect information. In this chapter we will not discuss these fads or other misunderstandings concerning OSS. However we will endorse you in this chapter with solid arguments that can help you when you are faced with fads regarding the use of open source solutions for security and privacy.

Some people are keen on ready to use list of good practices. However the context of security and privacy is very complex (organization, processes, people, technology). So we will not give a list of good practices. There are bad practices, but the list of good practices is almost unlimited, since the context for a random use case depends on various business aspects like:

- IT Knowledge and experience present in an organization.

- Information security knowledge present in an organization.
- Budget limitations.
- Time constrains.
- Maturity of IT within an organization. Especially with aspects like contracting, sourcing and IT operational management.
- Legal and regulatory aspects.

Depending on the exact needs and problems of an organization the way quality aspects for security and privacy solutions should be approached differs.

The following sections of this chapter covers the following questions:

- What is open source?
- Why should open source products be used for security and privacy solutions?
- What quality levels are needed for open source security and privacy products?
- What aspects are important when selecting security or privacy products for a solution architecture or within use in an organization?

What is open Source Software (OSS)?

Before even considering using open source products for security and privacy applications, it is strongly recommended that a good solid knowledge exist what open source really is.

In brief open source software is computer software for which the source code is available.

More in depth it is recommended to read the full definition of open source as provided by Open Source Initiative (http://opensource.org/osd):

1. Free Redistribution: The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.
2. Source Code: The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction

cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a pre-processor or translator are not allowed.

3. Derived Works: The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code: The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups: The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavour: The license must not restrict anyone from making use of the program in a specific field of endeavour. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License: The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product: The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software: The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral: No provision of the license may be predicated on any individual technology or style of interface.

Reading this long definition can you make confused. Especially when you need a shorter definition to explain to senior management the benefits of what open source is all about.

Open source is based on three concepts:

1. A development methodology that defines a community approach to developing software, meritocracy of developers, and quality based on peer review.

2. A licensing approach that provides free access to source code, conforms to one or more “Open Source Initiative” licenses, and prioritizes the rights of users and committers.

3. A community of users and developers with open participation.

Currently open source software is software that is licensed under one of several accepted free software or open source licenses approved by the Open Source Initiative that:

- do not restrict your ability to run the software, for any purpose,
- provide one with access to the source code,
- permit one to modify the software,
- permit one to share verbatim copies of the software with others, and
- under certain conditions, allow one to share one’s modifications with others.

"Open source software" is sometimes also called "Free software", "libre software", "Free/open source software (FOSS or F/OSS)", and "Free/Libre/Open Source Software (FLOSS)". The term "Free software" predates the term "open source software", but the term "Free software" has been sometimes misinterpreted as meaning "no cost", which is not the intended meaning in this context. ("Free" in "Free software" refers to freedom, not price.) So e.g. the free antivirus software AVG (http://www.avg.com) is no OSS software. In September 2015 Security firm AVG announced it will sell search and browser history data of users to advertisers in order to "make money" from its free antivirus software. Due to the fact that AVG is no OSS software, users who care about their privacy have no other choice than to look for an alternative antivirus package. If AVG was OSS software, presumable a software fork was created.

“Free software” means software that respects users' freedom and community. Roughly, it means that the users have the freedom to run, copy, distribute, study, change and improve the software. Thus, “free software” is a matter of liberty, not price.

The word "free" has many different meanings, and these different meanings often make it harder to understand OSS. The term "Free software" (as used in literature) is based on the

word "freedom" (the word "libre" is used in some other languages). However, "free" can also mean "no cost", and sometimes "no cost" products come with a "catch" that in fact is the opposite of freedom. A catch everyone in the IT knows as vendor lock in or (unhealthy) dependency.

To understand the concept of free, one should think of “free” as in “free speech,” not as in “free beer”. Sometimes OSS is called ‘libre software’ to show we do not mean it is gratis. A LinuxToday posting found a simple way to express these different meanings of the word free, which I'll slightly paraphrase here:

Free can mean various things:

- Free, as in free speech.
- Free, as in free beer.
- Free, as no cost.
- Free, as high on drugs

They are not all the same.

Free software(FOSS): Richard Stallman's Free Software Definition, adopted by the Free Software Foundation (FSF), defines free software as a matter of liberty, not price.

So summarized: Open source software (OSS) has nothing to do with no cost or no value. The initial cost structure for acquiring OSS based solutions is different. A license fee for the software use is absent. However to keep your solution supported by a vendor most companies pay a regular maintained fee to keep quality ask risk as low as possible. This is equal as with closed software solutions.

The power of open for security and privacy

To make improve security and privacy within digital worlds a number of aspects are of crucial importance:

- Open collaboration: This means that everyone can reuse and/or improve security and privacy related material (e.g. documentation).
- Use of open solutions: This means the application of OSS products for more and more security and privacy services. Many papers and books are written of the business advantage of using OSS software. When it comes down to security the main principle to go for OSS is openness. Using open solutions makes the solutions in the end more resistant against vulnerabilities. In the end it is about

transparent facts and quality criteria everyone can evaluate if needed. With closed source solution validation of quality statements is often not for all stakeholders possible. Think about the use of simple encryption software: We have more trust in an open encryption solutions that one that is claimed by a company that is unbreakable.

- Learn from each other and from our mistakes. People make mistakes. We make bad designs that increases security problems instead of solving them. OSS projects are not always managed as they should be when they produce critical security software. Learning in an open collaborative way without any direct or indirect commercial interest is crucial to get security and privacy aspects in IT where they should be: Just some quality criteria within the whole range of important aspects. In future the emphasis on security and privacy is equal as on safety, usability and business continuity. Currently only for safety aspects mandatory policies exist for companies to prevent people dying from software bugs. But today security and privacy aspects are not handled in the same way as safety aspects. A different approach is taken when it comes to designing IT systems on which human lives depend compared to designing information and privacy aspects in (business) information systems.

Improvements will not come overnight and a paradigm shift is needed for many companies to be more open and transparent regarding their security and privacy designs. Since privacy data is a core asset of customers of all companies, in future customers will demand a full transparent view on how a company protects the value assets given by customers.

Open security can be defined as an approach to use existing open knowledge in combination with the application of open source software (OSS) to help solve cyber security problems. OSS approaches collaboratively develop and maintain intellectual works (including software and documentation) by enabling us to use them for any purpose, as well as study, create, change, and redistribute them (in whole or in part).

Cyber security problems are created by starting with bad architecture or design or simply by a lack of knowledge and experience. Using an open security approach the security can be improved through collaboration.

So why use open source software for security and privacy applications? Open source software provides additional trust by allowing people to look into the source code whereas good OSS projects are completely transparent on all their SDLC and quality processes. When using OSS adjustments or improvements are easily made providing you with a flexible solution for your business.

Summary: Open source for use in the field of security and privacy means easy reuse (code or ideas), to improve what is already there. Reuse would be in a way so everyone can benefit. That way the quality gets better and better.

Determining quality of OSS for security and privacy applications

What quality really is or not has been a long running debate in many (scientific) management books. So it is only logic that quality in open source has been also a long running debate. However the fads regarding OSS made these discussions even harder to get a clear view on what should be defined as quality in relation to OSS security and privacy products. If you are planning to join these discussions, we would like to warn you to beware that these discussions are biased with many fads and unproven facts. Also many opinions in this field take an almost religious turn. General statements and general discussion seldom lead to weighted balanced judgment. IT for business use or security is not only the field of scientific computer science. Social sciences play a great role within IT security and privacy (think of the many awareness campaigns), and the field of risk management is not only the field of statisticians and mathematicians, but also psychology plays a role.

In essence the definition of quality and good OSS quality largely depends on the goal and context of the specific use case.

Quality and trust are for security and privacy products one of the most important aspects. This section will give guidelines on how quality of open source software for security and privacy can be easily measured and judged depending on your goal and use case.

Determination of the quality of security and privacy for a specific use case is complex. Besides an approved OSS licensed (see http://opensource.org) an OSS security products requires far more quality aspects. A license alone is not enough. This section describes a checklist to assist in evaluating the quality of an OSS products targeted on security and privacy. OSS products should always be evaluated on quality before use for real. But security and privacy OSS products have the following points that make evaluating a bit different:

- Trust

- Security (Unfortunately many security products are insecure and require insecure configuration to be usable!)

- Maintenance. Due to the SSL Heartbleed bug (http://heartbleed.com/) maintenance of OSS security products has grown in importance.

- Safety aspects can be compromised if security and privacy aspects are not handled well. Recent examples are car-hacking and plane-hacking. Due to security flaws, the safety can be compromised if intruders get into a system. Also personal safety (where do people live that …) can be harmed if for example web shops are sloppy with personal data and order records. Criminals like list of persons who buy very expensive paintings online.

The use of Open Source Software (OSS) components is a viable alternative to Commercial Off-The-Shelf (COTS) security and privacy components. Since the quality of OSS products varies widely, both industry and the research community have reported several OSS evaluation methods that are tailored to the specific characteristics of OSS. We have performed a systematic identification and evaluation of many of these methods, and present in this section the factors that really make sense with respects to:

- The endless types of context specific organizations that potentially use OSS security and privacy products.

- Protect (very)small and large security and privacy OSS projects who have very high product quality, but score less on (visible) process quality aspects.

- The variety in which security and privacy OSS products can be used within a SDLC.

The latest and most promising project for potential users to get a fast insight in OSS security projects is the “Core Infrastructure Initiative Best Practices Badge” project of the linuxfoundation.org. Badge will hopefully give in future some indication on some quality aspects regarding OSS security products. However the badges project has a specific scope and not all value reusable OSS software and projects are able to gain a badge. But also if an OSS has a badge, it still will be important to evaluate the use and risks for your use case.

A good security and privacy product should at least be evaluated on:

- Product quality aspects;

- Process quality aspects and

- Quality control system used to preserve product and process quality

In order to cut the complexity and not write endless notes on what quality is and how it can be measured we will focus in this section on given ready-to-use evaluation criteria. Use, reuse , or improve them. We will also try to collaborate with the badges project and similar OWASP projects to get one open evaluation list in future that is easy to use.

Note that some evaluation criteria are more important than others, but since quality is always context related evaluating the many different aspects further in depth should be done in a context specific solution architecture, not in this (general) reference architecture.

To keep things organized we use:

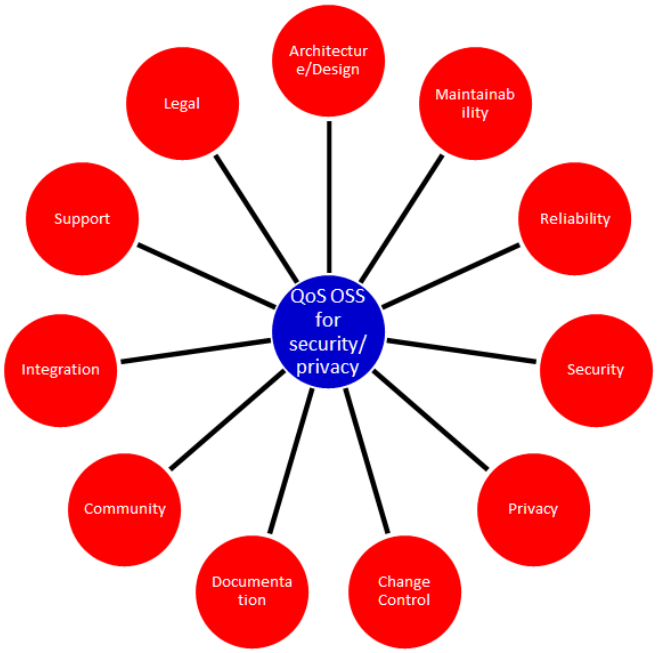
- ISO 25010 standard for software product quality (successor of the ISO 9126 standard)
- ISO/IEC15288 System Lifecycle Processes.

Note that ISO 25010 lacks attention for aspects like:

- Functional requirements
- Compliance (e.g. with laws, standards) requirements
- Documentation, Support and Training requirements

To overcome these aspects, we use our security and privacy principles in order to get an in-depth list of criteria that can be used for evaluation.

The following evaluation model is used:



Our main goal is to present in this reference architecture a list of evaluation criteria as simple as possible. So we enriched the criteria with simple (example) questions.

In the following paragraph key questions are given that can be used to evaluate an OSS security or privacy application for your use case.

Architecture and design

OSS projects that produce security or privacy software, solutions, libraries etc. should have:

- Defined principles.
- Defined requirements.
- Make reuse of e.g. good security and privacy standards to avoid reinventing the wheel.
- Readable architecture or design. So also people who are not programmers can understand the design. At least all design decision should be documented.

Unfortunately good security or privacy architectures and designs are rare for IT projects. This does not only account to large governmental projects, but also for large OSS security projects. Mind also that a big OSS security or privacy project can mean different things:

- Large number of users of a product or

- Enormous amount of source code
- Significant number of full time maintainers (over 10 is already a huge amount)
- Enormous number of contributors to a project.
- Etc.

E.g. the OpenSSL project has many users worldwide, however since the number of active project members was dramatically small, large is no guarantee for sustainable good quality.

Maintainability

When using OSS software you must have a strategy and a process that handles the maintenance of the software. Maintenance is essential for security and privacy related software products.

Maintenance has many aspects. For a healthy OSS security and privacy application you can divide maintenance in:

1. Maintenance on the OSS software product itself;
2. Maintenance on the quality system built around the eco system (processes, organization, financial s, control procedures, contributors and maintainers stability, etc.)
3. Maintenance process required for using the product.

Since this section only covers guidelines for evaluation of quality aspects of OSS security and privacy products we will only deal with the maintenance aspects directly related to the OSS product and organization surrounding it. But please beware: The maintenance required to be organized by you or your organization can differ significantly per OSS product. Some OSS security and privacy projects are aimed at making maintenance processes needed within your organization as simple as possible where other projects require more effort. Critical evaluation questions are:

- Is there a transparent way for (new)requirements adoption?
- Is there a strict change management process?
- Is there a tough release scheme? (A release early, release often (sometimes abbreviated RERO) approach). E.g. every month a new release.
- Is there a stable release and an alfa or beta release with new features?

- Is there an active community of developers?
- Are security vulnerabilities fixed in a structured way?
- Is there a source code release and a binary code release?
- What is the frequency of updates for the OSS project?
- Does the project use a build system that can automatically rebuild the software?
- Is there an automated test suite available?
- Are new tests always added for new functionality? (E.g. due to a internal policy?)

Maintainability plays a special role for open source cryptographic software algorithms. Cryptographic software requires next to excellent programming skills deep knowledge of cryptography. To be able to maintain cryptographic software finding the right resources is very hard. Within the security principle section some principles can be found that relate to quality aspects formulated for cryptographic software.

Reliability

Whenever you use an OSS security or privacy product you rely on protection or functionality. Reliability is a core aspect when evaluating OSS security and privacy products. Critical evaluation questions for reliability are:

- Is there an automatic test suite for the product?
- Does the testing methodology include (automatic) regression tests?
- Are interfaces with other products and platforms tested?
- Is there a written test plan along with documented test results?
- Are test reports published on the website?
- Is the software tested (when relevant) against OWASP top 10 vulnerabilities?
- Is the OWASP Application Security Verification Standard (ASVS) met?
- Is there a public accessible defects database?
- Is there a process organized around defects management?

- Is there a standard procedure followed before release new software in stable versions?
- Is the quality process documented?
- Is an endurance test under stress load performed with the public released version? Is this test public so everyone can (re)use it? (note: Not for all applications relevant)
- Has the project a website with a static URL?
- Is it clear who are project members, contributors and committers?
- Does a written procedure exist on how one can get commit rights on the core repository?
- Is there a public audit log available of changes on the core repository? (Subversion, Git and many other SSCM systems provide this crucial feature.)
- Are the SANS Securing Web Application Technologies (SWAT) criteria met?

Security

When using an OSS product you trust it is secure. Security is of course about trust, but when you use OSS security and privacy tools you must evaluate some crucial security aspects.

Unfortunately many security products exist that decrease your security. Software that requires insecure configurations for example or many nonstandard network sockets is not a good example of decent security.

Even if you are only testing a product or evaluating, you must have some criteria in place to prevent downloading malware or worse.

Some critical questions to determine some security aspects are:

- Are security vulnerabilities fixed according to a described process?
- Does the project have its own security officer or security team?

- Does a procedure exist and is it followed for performing a static and dynamic security code review on every major release? Are results of the secure code reviews available?

- A dynamic analysis tool for the code is used before releasing a major version (e.g. the project may use a fuzzing tool (e.g., American Fuzzy Lop) or a web application scanner (e.g., OWASP ZAP or w3af).

- What kind of socket connections and protocols have been used? Standard sockets connections (22,443,80,445) and standard protocols used (e.g. HTTPS, SSH, SSL, LDAP, LDAPs)?

- Are product vulnerabilities mentioned in the CVE database?

- Is it clear how many vulnerabilities (open and fixed) are mentioned in the CVE database? (Use the http://web.nvd.nist.gov/view/vuln/search?execution=e2s1) and search on product name. Note that vulnerabilities can be reported on the core product, but also on additional contributed modules if you are investigating a large OSS project.

- Is there a process to deal with vulnerabilities (e.g. release of fixed/patches in a controlled manner).

- Has the project created its own cryptographic libraries? Note that writing cryptographic algorithms is very hard and should be prevented by using already available good OSS algorithms.

- If cryptographic protocols or libraries are used, have these algorithms been published and reviewed by experts?

- Security and privacy principles and requirements are defined for the project and within the design and implementation it is clear how these are covered.

- All vulnerabilities are reported on the project site and are accessible without limitation by the public.

- It is clearly documented what process must be followed to obtain change rights on the main software repository.

- Procedures and policies exist to protect the code base from vandalism.

- Is a software release signed by a hash (minimal sha1 or stronger)?

Privacy

When you use an OSS security or privacy product you should not be required to register your name and organization in a database if it only serves a marketing purpose. All OSS licenses are very clear on what is allowed regarding distribution. People may sell OSS software. Even the GPL allows this. But since privacy aspects are becoming more and more important you must be aware on critical aspects that can harm your privacy when using OSS security or privacy tools.

Some critical questions to determine and evaluate privacy aspects are:

- The project has a clear written privacy policy on the website.

- Tracking cookies and other finger printing techniques are not used on the project core community website.

- The OSS security and privacy project respects the privacy of its users and contributors in all possible ways.

- Project maintainers and contributors are allowed to participate under an alias since not all governments allow working on OSS privacy or security products.
- The project is clear on measurements for handling contributors' personal identifiable data.
- Privacy of users or companies using the product is neither exposed nor stored.
- No privacy related data is stored and used by the project.

Change control

There can be no progress without change and if change is not taking place the bit rot will start. For security and privacy OSS projects some change control LCM aspects are of crucial importance. To make implementation of changes easy more and more projects enable an automatic update service that automatically implements changes on all running software instances. However implementing such a mechanism requires a very high level of internal change and governance processes.

Some questions to determine and evaluate change control aspects are:

- Has the project a version-controlled source repository that is publicly readable?
- Is issue tracking for defects in place? (For reporting bugs or feature request).
- Is tracking of requirements or enhancement on requirements request in place?
- Does the project release software with unique version numbering?
- Is a change log in human readable format for each release available?
- Does a clear documented SLCM process for the project exist?
- Is it clear how automatically built CI environments are configured and maintained?
- Does the change control process allow roll backs of releases?

Documentation

Software source code is not uniquely readable. Not everyone is a programmer and there is a huge number of dialects. Software code can be for example GO, Java, C/C++,PHP, Perl, Python, Javascript, Erlang, Scale etc. To be able to use software, configure it and get a quick impression of the quality of the project documentation is crucial. A project with good and solid documentation provides trust. Large popular OSS security and privacy projects will have many (commercial) books available. Good documentation creates good projects. Bad or not maintained documentation can kill a project.

Some questions to determine and evaluate documentation aspects to investigate the quality of an OSS security and privacy application are:

- Is documentation for new developers available for free on the website?
- Is the source code documented?
- Is documentation maintained?
- Does a structured written procedure exist on how the documentation is maintained?
- Are documentation processes embedded in the CI pipeline?
- Are the user manuals provided by the project?
- Is it directly clear what the status of the documentation is? Programmers usually do not write the user documentation. But it is crucial that the documentation keeps in sync with every release.
- Are there (many) books (besides the one published by the project itself) available?
- Is commercial documentation available (e.g. books on Amazone)?
- Can everyone participate in improving the documentation?
- Is the documentation published under a Creative Commons licenses (CC) license?

Community

All solid OSS security and privacy projects have a strong and stable community. By evaluating community aspects one can get an indication on how the project deals with all kind of crucial quality aspects on product level and on process level. A community does not have to be large and very active. Many good security projects exist with 2-3 community

members who manage to perform all crucial processes on a periodic basis. Stability is often more important than size. An OSS project that has too many forks can be an indication of a strong vision of the leader or a lack of leadership on dealing with crucial issues regarding the project health. A fork is most of the time a good sign. It means the software is used in many different ways and some people are building other communities to support their future vision for the project. But some research on why a project is forked should be done when you are evaluating OSS security products that offer exact the same functionality and share the same code base.

Some questions that can assist you in evaluation community related aspects:

- How big is the community of core developers?
- Is the process of joining the OSS project transparent?
- Is it clear how one can become a code submitter?
- Is the process around the core community open and transparent?
- Are commercial books of the project available?
- What is the number of available commercial books of the project?
- Are many books available? (E.g. on amazone.com or O'Reilly)
- Are mailing lists of the core developers open and transparent?
- Is it clear how decisions are made within the project?
- Can everyone attend to all project discussions (e.g. mailing list, slack channels, IRC)?
- Average number of people active on IRC or slack?
- The project has a written policy to stay active and healthy (e.g. the C4, see zmq)

Integration

Using OSS security or privacy software is always done in a specific context. You already have other software building blocks, you need your own reports, or you want to use another identity manager to use the product. Integration aspects on business and technical level are crucial for healthy OSS security and privacy projects. Too often projects fall victim to scope creep and are creating a one-size-fits-all solution. Logging, auditing and encryption e.g. are services are a world of themselves. The same goes for great responsive

GUI's. You cannot create an excellent CMS when you are focusing on a dedicated security or privacy function.

Some questions that can help you evaluate integration aspects of OSS security and privacy products:

- Can the software easily be integrated with non OSS or other OSS projects?
- Does the software allow an easy way to extend its functionality?
- Is the software modular built?
- Are REST interfaces available?
- Are all interfaces for external use stateless by design?
- Are API's well documented?
- Does the OSS license have impact on building your own library or module against the core product? E.g. the GPL is very clear on integration.
- Is it clear how security or privacy aspects are impacted when third party integration modules are used?

Support

Every organization using OSS security and privacy products sooner or later needs some professional support to maintain the product, to adjust configuration settings or to implement new versions. Within many businesses support on software is crucial and it is often written down in lengthy support contracts with many sentences that must make clear what kind of support is requested. In general, when you have a good relationship with a company that supports some (OSS) software for you, the contract should be based more on trust. Lengthy contracts are usually the result of little confidence or expensive mistakes made in the past. The great advantage of using OSS security and privacy products is that you can be very flexible in how you organize crucial support issues for a product. Of course when you rebuild the product it will be hard to find people who can easily resolve problems. Some OSS security and privacy products have a commercial version for which you can get paid support. But when the commercial version differs from the OSS version you are not dealing with a healthy OSS project anymore.

A large and well known OSS security and privacy project has many excellent people within the community who are willing to provide support.

Some questions that help you evaluate support aspects regarding OSS security and privacy products:

- Is paid support possible?
- Is there a strong community support?
- Can questions on usage, configuration or problems be posted somewhere?
- Has the project an active open forum or mailing list for support questions?
- Does a mailing list exist for paid support or contracting work corporate users of the product?
- Is it possible to contact one of the core developer(s) working on the product directly (e.g. email?)

Legal

Security and privacy application can have many legal aspects. This applies not only to the usage, but also to the possession and creation of security and privacy related software. Many governments suspect people who use encryption software for private use. In some countries the use of privacy protection tools is prohibited. When using OSS security and privacy products it can be crucial to evaluate the legal aspects first, before using the product. Many security or privacy OSS products are great tools for criminals. This cannot be avoided. When someone uses a tablet to smash people on the head Apple cannot be accused of creating a murder weapon. However responsible projects are aware of possible trivial misuse.

Some questions that can help you evaluate legal aspects of OSS security and privacy products:

- Which OSS license is used?
- Is the license approved by the OSI foundation?
- Is the OSS license a widely used license?
- All functionality must meet the *Open Standards Requirement for Software* by the Open Source Initiative

- Is the OSS project aware of any possible misuse of the product? E.g. does a notice exist on what the intention for correct usage is of the product?

- Can you be held responsible for damage or lawbreaking when you use the product on the open internet? Does the project warn you for this kind of aspects?

OSS Security and Privacy System Building Blocks

Introduction

When you know the advantages and disadvantages of using open source building blocks for your security architecture or design, this chapter provides an overview on some open source security solutions. When you have all your security/privacy principles, requirements, attack vectors and security persona's clear the hardest task is to select (or create) solution building blocks that covers the needed functionality. Using OSS security or privacy Solution building Blocks within your solution architecture can give significant advantages. See section "The power of open for security and privacy" in this reference architecture for more information on the advantages.

We know we can never be complete with an overview of OSS security and privacy applications. The overview in this chapter is created end of 2015 and is just a guidance to give you:

- Insights on what type of products are available in the OSS domain.
- A collection of OSS solution building blocks for your security architecture or design you can consider to evaluate for your specific use case.
- Some ideas of solutions you are perhaps not familiar with.

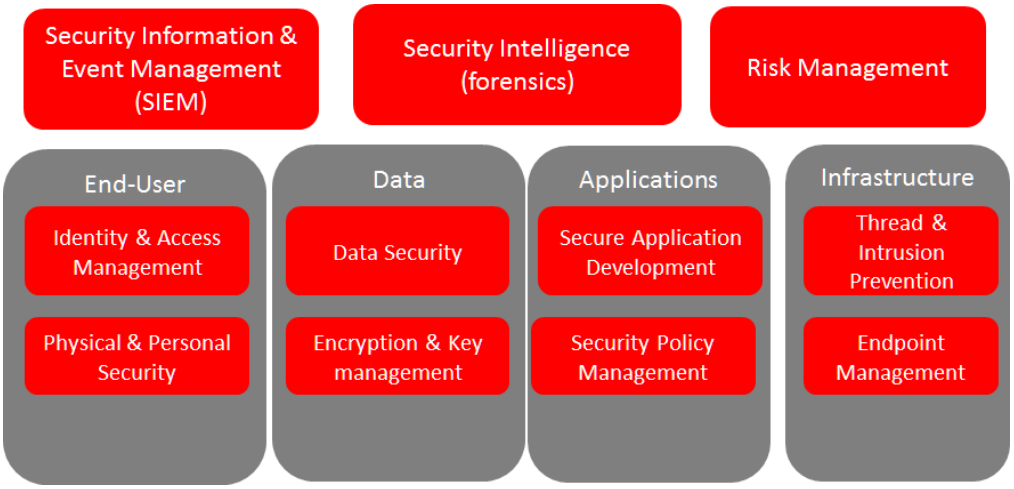
There are now a million different open source software projects published somewhere on the internet. Our holy grail is to keep track of the top 50 security and privacy open source projects for every security and privacy service needed within a business architecture. This way when you need a secure logging service you can evaluate the top 50 projects first before searching further or creating (aka forking) your own. In this first release of this OSS Security and Privacy reference architecture we yet are far away from this goal.

Criteria used for products mentioned in this chapter are:

- The products must have a valid OSS license;
- The project must be active and must meet a certain quality level;
- The product must be in use somewhere (*)

(*) Unfortunately we can and never will expose information where products are in use, however many mature products have solid references on their website, along with active user groups.

The number of OSS security and privacy applications available is over overwhelming. Using the following conceptual topology can help with arranging functional to product mapping needs:



For every security or privacy function or service needed you should look serious at using open transparent reusable solutions. So Open Source. Of course many vendors provide good solid security products for specific use cases. But when you feel you need a trivial security or privacy service, there is almost always a working and maintained OSS application available. When using an OSS solutions, you should have a large choice of companies that deliver maintenance and support on this application on commercial bases.

OSS Security Applications

American fuzzy lop

SBB Description

American fuzzy lop is a security-oriented fuzzer that employs a novel type of compile-time instrumentation and genetic algorithms to automatically discover clean, interesting test cases that trigger new internal states in the targeted binary. This substantially improves the functional coverage for the fuzzed code.

These tool can be very productive in determining security flaws: The famous SSL Heartbleed bug was found in record time using this software. See https://blog.hboeck.de/archives/868-How-Heartbleed-couldve-been-found.html.

SBB License GNU General Public License (GPL) 2.0

Core Technology C

Project URL http://lcamtuf.coredump.cx/afl/

Source Location http://lcamtuf.coredump.cx/afl/releases/

Tag(s) Security, Test Tool

Bokken (Open Source Reverse Code Engineering)

SBB Description

Bokken is an Open Source Reverse Code Engineering tool.

Bokken is a **GUI for the Pyew and Radare projects** so it offers almost all the

	same features that Pyew has and and some of the Radare's ones. It's intended to be a **basic disassembler**, mainly, to analyze malware and vulnerabilities. Currently Bokken **is neither** an hexadecimal editor **nor** a full featured disassembler **YET**, so it should not be used for deep code analysis or to try to modify files with it. Bokken has the ability to detect and analyze **PE/Elf/mach0** files so, when one of those file formats is detected, the GUI shows all the information found on the analysis and offers many additional options to study the file.
SBB License	GNU General Public License (GPL) 2.0
Core Technology	Python
Project URL	http://bokken.re
Source Location	https://inguma.eu/projects/bokken/repository
Tag(s)	Code Analyzer, Security

Bosun

SBB Description	Bosun is an open-source, MIT licensed, monitoring and alerting system by Stack Exchange. It has an expressive domain specific language for evaluating alerts and creating detailed notifications. It also lets you test your alerts against history for a faster development experience. Collecting metrics about our systems is fun but what makes a monitoring system useful is alerting when anomalies arise. This is the real strength of Bosun.

	Bosun encourages a particular workflow that makes it easy to design, test, and deploy an alert. If you look at the top of the Bosun display, the tabs include Items, Graph, Expression, Rule, and Test config in left-to-right order; that reflects the phases you go through as you create an alert. In general, first you'll select an item (metric) that is the basis of the alert.
SBB License	GNU General Public License (GPL) 2.0
Core Technology	GO
Project URL	http://bosun.org/
Source Location	https://github.com/bosun-monitor/bosun
Tag(s)	Security, SIEM

Bro

	Bro is a powerful network analysis framework. Bro is a passive, open-source network traffic analyzer. It is primarily a security monitor that inspects all traffic on a link in depth for signs of suspicious activity. Bro supports a wide range of traffic analysis tasks even outside of the security domain, including performance measurements and helping with trouble-shooting.
SBB Description	The most immediate benefit that a site gains from deploying Bro is an extensive set of *log files* that record a network's activity in high-level terms. These logs include not only a comprehensive record of every connection seen on the wire, but also application-layer transcripts such as, e.g., all HTTP sessions with their requested URIs, key headers, MIME types, and server responses; DNS requests with replies; SSL certificates; key content of SMTP sessions; and much more. By default, Bro writes all this information into

	well-structured tab-separated log files suitable for post-processing with external software. Users can however also chose from a set of alternative output formats and backends to interface directly with, e.g., external databases.
SBB License	GNU General Public License (GPL) 2.0
Core Technology	C
Project URL	https://www.bro.org
Source Location	https://github.com/bro
Tag(s)	IDS, Security

Data Seal

	Data Seal is a lightweight, UELMA-compliant data authentication service. Data Seal is a project of U.S. Open Data to provide a system where open data released by governments can be authenticated by end users—whether or not the data was most recently downloaded from the official source.
SBB Description	Government data releases need to abide by local laws (for example, the District of Columbia Official Code) and should also abide by the Uniform Electronic Legal Material Act (UELMA). Part of the UELMA provisions state that "legal material be…authenticated, by providing a method to determine that it is unaltered". Data Seal provides agencies with a web-based interface to provide this functionality.

SBB License	GNU General Public License (GPL) 2.0
Core Technology	Django/Python
Project URL	https://github.com/unitedstates/data-seal/wiki
Source Location	https://github.com/unitedstates/data-seal
Tag(s)	data authentication, Security

FIDO (Fully Integrated Defense Operation)

SBB Description	FIDO (Fully Integrated Defense Operation – apologies to the FIDO Alliance for acronym collision) is developed by NetFlix and now OSS. This system is for automatically analyzing security events and responding to security incidents. The premise of FIDO is simple… each year companies are receiving an ever increasing amount of security related alerts. Instead of hiring more analyst to comb through the endless stream of alerts we automate the analysis to combat the barrage of information. Simply put, we integrate and then automate the manual human processes by codifying the logic and process used by threat analysts to provide consistent and reliable results. The typical process for investigating security-related alerts is labor intensive and largely manual. To make the situation more difficult, as attacks increase in number and diversity, there is an increasing array of detection systems deployed and generating even more alerts for security teams to investigate. FIDO is a NetFlix OSS project, see: http://techblog.netflix.com/2015/05/introducing-fido-automated-security.html

SBB License Apache License 2.0

Core Technology C#

Project URL https://github.com/Netflix/Fido/wiki

Source Location https://github.com/Netflix/Fido

Tag(s) Security, SIEM

Gryffin

SBB Description Gryffin is a large scale web security scanning platform. Created by Yahoo, and since September 2015 available as open source.

It is not yet another scanner. It was written to solve two specific problems with existing scanners: coverage and scale. Better coverage translates to fewer false negatives. Inherent scalability translates to capability of scanning, and supporting a large elastic application infrastructure. Simply put, the ability to scan 1000 applications today to 100,000 applications tomorrow by straightforward horizontal scaling.

SBB License MIT License

Core Technology Go

Project URL https://github.com/yahoo/gryffin

Source Location	https://github.com/yahoo/gryffin
Tag(s)	IDS, Security, Vulnerability scanning

Kali

SBB Description	Kali is the most complete 'Penetration Testing Linux Distribution' around. Everything you need for penetration testing is collected, tested and made available on this linux distribution. Of course all tools are OSS. The complete list of tools can be found here:http://tools.kali.org/tools-listing
SBB License	GNU General Public License (GPL) 2.0
Core Technology	N.A. (OSS Tool collection)
Project URL	https://www.kali.org/
Source Location	http://git.kali.org/gitweb/
Tag(s)	Security, Sniffer, Vulnerability scanning

Kismet

SBB Description	Kismet is an 802.11 layer2 wireless network detector, sniffer, and intrusion detection system. Kismet will work with any wireless card which supports raw monitoring (rfmon) mode, and (with appropriate hardware) can sniff 802.11b, 802.11a, 802.11g, and 802.11n traffic. Kismet also supports plugins which allow sniffing other media such as DECT. Kismet identifies networks by passively collecting packets and detecting standard named networks, detecting (and given time, decloaking) hidden networks, and inferring the presence of non beaconing networks via data traffic. The great feature of Kismet is that this tool works working passively, so detection by IDS is prevented when scanning WLAN's.
SBB License	GNU General Public License (GPL) 2.0
Core Technology	C++
Project URL	http://www.kismetwireless.net/
Source Location	https://www.kismetwireless.net/code/
Tag(s)	IDS, Security, Sniffer

Libreswan

SBB Description	Libreswan is an IPsec implementation for Linux. Libreswan is a free software implementation of the most widely supported and standarized VPN protocol based on ("IPsec") and the Internet Key Exchange ("IKE").

SBB License	GNU General Public License (GPL) 2.0
Core Technology	
Project URL	https://libreswan.org/
Source Location	https://github.com/libreswan/libreswan
Tag(s)	communication, Cryptography, Security

Lynis

SBB Description	Lynis is a suite of tools (shell scripts) for security auditing, compliance and hardening for Linux, Mac OS, and Unix based systems. Of course many (better) audit tools are available, but this one is simple and straightforward. So easy to extend and to improve. Especially if you like shell-scripting. Michael Boelen from the Netherlands (owner of company cisofy.com) created this software.
SBB License	GNU General Public License (GPL) 2.0
Core Technology	unix-shell scripts
Project URL	https://cisofy.com
Source	https://github.com/CISOfy/lynis/

Location

Tag(s) Audit, Security

Mantra

OWASP Mantra is a collection of free and open source tools integrated into a web browser, which can become handy for students, penetration testers, web application developers,security professionals etc. It is portable, ready-to-run, compact and follows the true spirit of free and open source software.

Mantra is lite, flexible, portable and user friendly with a nice graphical user interface. You can carry it in memory cards, flash drives, CD/DVDs, etc. It can be run natively on Linux, Windows and Mac platforms. It can also be installed on to your system within minutes. Mantra is absolutely free of cost and takes no time for you to set up.

SBB Description Mantra is a browser especially designed for web application security testing. By having such a product, more people will come to know the easiness and flexibility of being able to follow basic testing procedures within the browser. Mantra believes that having such a portable, easy to use and yet powerful platform can be helpful for the industry.

Mantra has many built in tools to modify headers, manipulate input strings, replay GET/POST requests, edit cookies, quickly switch between multiple proxies, control forced redirects etc. This makes it a good software for performing basic security checks and sometimes, exploitation. Thus, Mantra can be used to solve basic levels of various web based CTFs, showcase security issues in vulnerable web applications etc.

SBB License GNU General Public License (GPL) 3.0

Core Technology javascript

Project URL http://www.getmantra.com

Source Location https://code.google.com/p/getmantra/

Tag(s) Security, Test Tool

OpenVAS

SBB Description OpenVAS is a framework of several services and tools offering a comprehensive and powerful vulnerability scanning and vulnerability management solution.

The core of this SSL-secured service-oriented architecture is the **OpenVAS Scanner**. The scanner very efficiently executes the actual Network Vulnerability Tests (NVTs) which are served with daily updates via the OpenVAS NVT Feed or via a commercial feed service.

SBB License GNU General Public License (GPL) 2.0

Core Technology C

Project URL http://www.openvas.org

Source Location https://scm.wald.intevation.org/svn/openvas/trunk

Tag(s) Security, Vulnerability scanning

OWASP ZCR Shellcoder

SBB Description	OWASP ZCR Shellcoder is an open source software in python language which lets you generate customized shellcodes for various operation systems. Shellcodesare small codes in assembly which could be use as the payload in software exploiting. Other usages are in malwares, bypassing antiviruses, obfuscated codes and etc.
SBB License	GNU General Public License (GPL) 3.0
Core Technology	Python
Project URL	https://www.owasp.org/index.php/OWASP_ZSC_Tool_Project
Source Location	https://github.com/Ali-Razmjoo/OWASP-ZSC/
Tag(s)	Security, Test Tool

OWASP Zed Attack Proxy (ZAP)

SBB Description	The OWASP Zed Attack Proxy (ZAP) is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications. It is designed to be used by people with a wide range of security experience and as such is ideal for developers and functional testers who are new to penetration testing as well as being a useful addition to an experienced pen

	testers toolbox.
SBB License	Apache License 2.0
Core Technology	Java
Project URL	https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project#tab=Main
Source Location	https://github.com/zaproxy/zaproxy
Tag(s)	Security

Phpseclib (PHP Secure Communications Library)

SBB Description	Phpseclib is designed to be ultra-compatible. It works on PHP4+ (PHP4, assuming the use of PHP_Compat) and doesn't require any extensions. For purposes of speed, **mcrypt is used** if it's available **as is gmp or bcmath** (in that order), but they are not required. Phpseclib is designed to be fully interoperable with OpenSSL and other standardized cryptography programs and protocols. Phpseclib is a pure-PHP implementations of: • BigIntegers • RSA • SSH2

- SFTP
- X.509
- Symmetric key encryption
  - AES
  - Rijndael
  - Twofish
  - Blowfish
  - DES
  - 3DES
  - RC4
  - RC2

SBB License MIT License

Core Technology PHP

Project URL http://phpseclib.sourceforge.net/

Source Location https://github.com/phpseclib/phpseclib

Tag(s) Cryptography, Security

RIPS (code analyser)

SBB Description

RIPS is a tool written in PHP to find vulnerabilities in PHP applications using static code analysis. By tokenizing and parsing all source code files RIPS is able to transform PHP source code into a program model and to detect sensitive sinks (potentially vulnerable functions) that can be tainted by userinput (influenced by a malicious user) during the program flow. Besides the structured output of found vulnerabilities RIPS also offers an integrated code audit framework for further manual analysis.

RIPS was released during the Month of PHP Security (www.php-security.org).

Features

- detect XSS, SQLi, File disclosure, LFI/RFI, RCE vulnerabilities and more
- 5 verbosity levels for debugging your scan results
- mark vulnerable lines in source code viewer
- highlight variables in the code viewer
- user-defined function code by mouse-over on detected call
- active jumping between function declaration and calls
- list of all user-defined functions (defines and calls), program entry points (user input) and scanned files (with includes) connected to the source code viewer
- graph visualization for files and includes as well as functions and calls
- create CURL exploits for detected vulnerabilities with few clicks
- visualization, description, example, PoC, patch and securing function list for every vulnerability
- 7 different syntax highlighting colour schemata
- display scan result in form of a top-down flow or bottom-up trace
- only minimal requirement is a local webserver with PHP and a browser (tested with Firefox)

- regex search function

SBB License GNU General Public License (GPL) 3.0

Core Technology PHP

Project URL http://rips-scanner.sourceforge.net/

Source Location http://sourceforge.net/projects/rips-scanner/

Tag(s) Code Analyzer, Security

Security Monkey

SBB Description Security Monkey monitors policy changes and alerts on insecure configurations in an AWS account. While Security Monkey's main purpose is security, it also proves a useful tool for tracking down potential problems as it is essentially a change tracking system.

More information: http://techblog.netflix.com/2014/06/announcing-security-monkey-aws-security.html

SBB License Apache License 2.0

Core Technology Python

Project URL http://securitymonkey.readthedocs.org/en/latest/

Source Location	https://github.com/Netflix/security_monkey
Tag(s)	Security, SIEM

SIMP (The System Integrity Management Platform)

	SIMP is a framework that aims to provide a reasonable combination of security compliance and operational flexibility. Fundamentally, SIMP is a framework that is designed to be secure from a practical point of view out of the box. As a framework, SIMP is designed to be flexed to meet the needs of the end user.
SBB Description	The ultimate goal of the project is to provide a complete management environment focused on compliance with the various profiles in the SCAP Security Guide Project and industry best practice. Though it is fully capable out of the box, the intent of SIMP is to be molded to your target environment in such a way that deviations are easily identifiable to both Operations Teams and Security Officers. This project is released to the public by the US National Security Agency.
SBB License	MIT License
Core Technology	
Project URL	https://github.com/NationalSecurityAgency/SIMP
Source Location	https://github.com/simp

Tag(s) Audit, Security

Simplify

	Simplify uses a virtual machine to understand what an app does. Then, it applies optimizations to create code that behaves identically, but is easier for a human to understand. Specifically, it takes Smali files as input and outputs a Dex file with (hopefully) identical semantics but less complicated structure.
SBB Description	For example, if an app's strings are encrypted, Simplify will interpret the app in its own virtual machine to determine semantics. Then, it uses the apps own code to decrypt the strings and replaces the encrypted strings and the decryption method calls with the decrypted versions. It's a **generic** deobfuscator because Simplify doesn't need to know how the decryption works ahead of time. This technique also works well for eliminating different types of white noise, such as no-ops and useless arithmetic.
SBB License	MIT License
Core Technology	
Project URL	
Source Location	https://github.com/CalebFenton/simplify
Tag(s)	Code Analyzer, Security

Streisand

SBB Description	Streisand is software for setting up secure connections with your friends. A bit like TOR. Communication can be sets up over L2TP/IPsec, OpenSSH, OpenVPN, Shadowsocks, sslh, Stunnel, and a Tor bridge.
SBB License	GNU General Public License (GPL) 3.0
Core Technology	Python
Project URL	https://github.com/jlund/streisand
Source Location	https://github.com/jlund/streisand
Tag(s)	Privacy, Security

Stunnel

SBB Description	Stunnel is a proxy designed to add TLS encryption functionality to existing clients and servers without any changes in the programs' code. Its architecture is optimized for security, portability, and scalability (including load-balancing), making it suitable for large deployments. Stunnel uses the OpenSSL library for cryptography, so it supports whatever cryptographic algorithms are compiled into the library. It can benefit from the FIPS 140-2 validation of the OpenSSL FIPS Object Module, as long as the building process meets its Security Policy.

SBB License	GNU General Public License (GPL) 2.0
Core Technology	C
Project URL	https://www.stunnel.org/index.html
Source Location	http://www.usenix.org.uk/mirrors/stunnel/
Tag(s)	Cryptography, Security

Suricata

SBB Description	Suricata is a high performance Network IDS, IPS and Network Security Monitoring engine. Open Source and owned by a community run non-profit foundation, the Open Information Security Foundation (OISF). Suricata is developed by the OISF and its supporting vendors.
SBB License	GNU General Public License (GPL) 2.0
Core Technology	C
Project URL	http://suricata-ids.org
Source Location	https://github.com/inliniac/suricata

Tag(s) IDS, Security

SWAMP (Software Assurance Marketplace)

SBB Description

This security application is a SAAS solution. However it is built of OSS building blocks and available to be use under an friendly OSS license for everyone.

- Capabilities of the SWAMP
- Static analysis
- Operates on the original source code
- Tracks problems down to the location in the original code
- Relatively quick and easy to use
- Provides complete code coverage
- Compare results from multiple tools
- Find and visualize overlaps
- Correlate results

Languages supported: C/C++,Java source, Java bytecode, Python, Ruby. PHP and Javascript are on the roadmap for end 2015 to be supported.

SBB License GNU General Public License (GPL) 3.0

Core Technology

Project URL	https://www.mir-swamp.org
Source Location	
Tag(s)	Code Analyzer, Security

Tor

SBB Description	Tor is free software and an open network that helps you defend against traffic analysis, a form of network surveillance that threatens personal freedom and privacy, confidential business activities and relationships, and state security. Creating your own Tor network is easy with this software, or use existing Tor nodes.
SBB License	GNU General Public License (GPL) 2.0
Core Technology	
Project URL	https://www.torproject.org
Source Location	https://www.torproject.org/dist/
Tag(s)	Cryptography, Privacy, Security

Vault

SBB Description	Vault is a tool for securely accessing secrets. A secret is anything that you want to tightly control access to, such as API keys, passwords, certificates, and more. Vault provides a unified interface to any secret, while providing tight access control and recording a detailed audit log. Vault secures, stores, and tightly controls access to tokens, passwords, certificates, API keys, and other secrets in modern computing. Vault handles leasing, key revocation, key rolling, and auditing. Vault presents a unified API to access multiple backends: HSMs, AWS IAM, SQL databases, raw key/value, and more. A modern system requires access to a multitude of secrets: database credentials, API keys for external services, credentials for service-oriented architecture communication, etc. Understanding who is accessing what secrets is already very difficult and platform-specific. Adding on key rolling, secure storage, and detailed audit logs is almost impossible without a custom solution. This is where Vault steps in.
SBB License	Mozilla Public License (MPL) 1.1
Core Technology	GO
Project URL	https://vaultproject.io
Source Location	https://github.com/hashicorp/vault
Tag(s)	Security

w3af (Web Application Attack and Audit Framework)

SBB Description

w3af is a Web Application Attack and Audit Framework. The project's goal is to create a framework to help you secure your web applications by finding and exploiting all web application vulnerabilities.

The w3af framework is divided into three main sections:

1. The core, which coordinates the whole process and provides libraries for using in plugins.
2. The user interfaces, which allow the user to configure and start scans
3. The plugins, which find links and vulnerabilities

SBB License GNU General Public License (GPL) 2.0

Core Technology Phython

Project URL http://w3af.org/

Source Location https://github.com/andresriancho/w3af/

Tag(s) Audit, Security, Test Tool

References

When creating this reference architecture, we performed serious research. We used many valuable sources (books, articles, scientific publications, blogs, etc). Below some reference for those who like to have more background information.

AICPA/CICA Privacy Maturity Model March 2011, http://www.aicpa.org/InterestAreas/InformationTechnology/Resources/Privacy/GenerallyAcceptedPrivacyPrinciples/DownloadableDocuments/AICPA-CICA-Privacy-Maturity-Model-ebook.pdf

Common Weakness Enumeration (CWE™), cwe.mitre.org

Generally Accepted Privacy Principles (GAPP),https://www.cippguide.org/2010/07/01/generally-accepted-privacy-principles-gapp/

Jericho security model, Open Group, https://collaboration.opengroup.org/jericho/

NIST, http://www.nist.gov/cyberframework/index.cfm

OECD privacy framework 2009, 2010,http://oecdprivacy.org/

Open Security Architecture (OSA), http://www.opensecurityarchitecture.org/

Open State Foundation, http://www.openstate.eu/

OSS Security Badges project (Work in progress), D. Wheeler, https://github.com/linuxfoundation/cii-best-practices-badge/blob/master/criteria.md

Privacy Management Reference Model and Methodology (PMRM) Version 1.0, Committee Specification Draft 01, 26 March 2012, http://docs.oasis-open.org/pmrm/PMRM/v1.0/csd01/PMRM-v1.0-csd01.pdf

Privacy Management Reference Model and Methodology (PMRM) Version 1.0, http://docs.oasis-open.org/pmrm/PMRM/v1.0/csd01/PMRM-v1.0-csd01.html

Securing Web Application Technologies [SWAT] Checklist , https://software-security.sans.org/resources/swat

Security in-a-box, Tactical Technology Collective and Front Line Defenders,https://securityinabox.org/en

Software Assurance Maturity Model (OWASP), http://www.opensamm.org/

The Free Software Foundation,https://www.gnu.org

The Open Source Initiative (OSI), http://opensource.org/licenses/

Web Authorization Protocol (OAuth), https://tools.ietf.org/html/draft-ietf-oauth-v2-threatmodel-01

Licensing

Thank you for downloading or buying this book. We want people to reuse content of this reference architecture in their own security solution architectures or privacy solution architectures. Security is hard enough, so reuse good open solutions available today. If you like to reuse text of this reference architecture in your own work: presentations, articles or your own book you are free to do so under the conditions that belong to the Creative Commons cc-by-sa license.

We have chosen to use the cc-by-sa license so this reference architecture is created to be shared as much as possible. Also using the cc-by-sa license lowers barriers for creating a better version of this reference architecture.

- You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation.

- No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

Contributing

We encourage all security professionals to improve this reference architecture. Join the team to:

- Add security or privacy principles.
- Add security or privacy models.
- Help us create the largest OSS reference framework on OSS security and privacy applications and tools.
- Create better graphics and text.
- Add threat models that can be easily reused.
- Improve criteria on selecting OSS solutions for security and privacy applications.
- Create tools to speed up the process of making use of this reference architecture. E.g. we created a GPL WordPress tool to manage and create security specification documents fast. Help us to improve these tools or create your own.

Your contributions to this Guide are greatly appreciated as long as contributions fit within the scope and goal of this security and privacy reference architecture. As an open project, this Open Reference Architecture for Security and Privacy shall always remain vendor-neutral and freely available for all to use. If you contribute you will of course get credit (mentioned in upcoming publications).

You can contribute using the following Github repository:

https://github.com/nocomplexity/SecurityPrivacyReferenceArchitecture

Please observe our contribution guidelines before creating a pull request:

With the exception of typos and spelling mistakes (feel free to fix these and they'll be merged), please observe the following guides:

- Always open an issue first. This will allow us to determine whether or not the change should take place. Explain your issue, and we will discuss it with you. If we agree the change is necessary we will mark it as TODO and will fix it when

we get a chance, or we will allow a member of the community to supply the change with a pull request.

- Note that this reference architecture is intended to be a helpful resource aimed at professional security/privacy architects and designers.
- Contributions must fit within the scope and goal of this security and privacy reference architecture. Of course we like to discuss your input for changing scope or goals if needed!

Please follow the following procedure when contributing to this document:

- Fork the chapter you want to change or contribute on GitHub, with the Fork button
- Clone the repository to your computer
- Create a branch in which you make your patch git checkout -b <branchname>
- Make your changes, commit and push the branch
    - edit, edit, edit
    - git add files, git commit
    - git push origin <branchname>
- Create a pull request for the branch <branchname> you created (not 'master')

Since we know many security professionals are not familiar with GitHub, we are currently investigating other methods to lower barriers for contributing to this project.

The maintainers review your pull request and your patch is merged with the master branch ASAP.

Licensing

When you submit text to which you hold the copyright, you agree to license it under:

- Creative Commons Attribution-ShareAlike 4.0 Unported License (“CC BY-SA”), or
- CC0 1.0 Universal (CC0 1.0)