Architectural Risk Analysis (Threat Modelling)
Based on the Software Security lecture by
Antti Vähä-Sipilä

Liisa Lado–Villar
Department of Computer Science
University of Helsinki
liisa.lado-villar@helsinki.fi

April 16, 2014

In this session we going to start with some discussion about terminology. Then there will be some discussion about business level security risk analysis, following by technical analysis which contains discovering security needs. Visualising system design and visualising possible point of threats will be discussed, this analysis by drawing each level of system, drawing diagrams and charts. Next topic is about security boundaries in diagrams. Where the data may get tampered, taint data will be considered. after analysing data flows there need to be think about security at every point of diagrams, CIA is introduced and STRIDE is discussed. Finally there will be discussion at last part of the findings and there meaning , risk analysis.

Threat and risk are here used interchangeably. Threat modelling and security risk analysis are used in the same meaning.

Risk analysis can be done at two level, business level and technological. Business threat model is about the value of the business including value of software. Here we concentrate merely architectural and design level, that is the technological level. In literature the meaning varies, take this to account. McGraw [2, 140] gives definition for risk analysis being as identifying an ranking at some particular stage in the software development lifecycle.

Graff & van Wyk [1, p. 30] say the process of selecting design elements and principles to match a defined security need is security architecture.

This chapter contains part of the most important topic of software security course.
Visualising With Threat Trees

For visualising there are threat trees or attack trees, where assets and threats are connected in tree shape in an image. This doesn’t express well modelled threats, because issues might have multiple causes and effects. Trees turn out to be more attack graphs (spaghetti), but excluding multiple arcs coming and leaving from a node. Doesn’t scale well, size tend to explode compared to size of target being analysed.

![Attack graph for 14-machine subnet. From the paper of Noel & Jajodia [3]](image)

Overview of Business Level Security Risk Analysis

Business value is important, business brings the money in. For backing is need to learn business value adding of the system, perhaps from product manager or business owner. Learn business level diagram then compare to technical, then decide risks.

Notice that, some things that may be technically security risks might not be business risks. The mitigation of these may not be necessary. Notice also that technical architecture may not alone reveal all threats. Do this analysis with business people in team. Business level security risk analysis contains
identifying information flows and assets. Don’t leave out things, that aren’t visible in your component but have an overall security effects, for example usage analytic data. Also notice personal data. Identify main use cases (user stories) and the misuse cases.

Identify all roles of people, like users, admins, sales, service, or support the product. Identify regulations (for compliance), customer requirements, certifications and their influence on software security.

The main takeaways that will help you later, are:

• The description of what is the system used for business wise.
• List of information assets and list of information flow.
• List of persons and their roles in interacting with the system.
• What is the top list of things that should not happen (i.e., misuse cases).

Technical Analysis: Discovering the Needs

Methodical security needs assessment as apart of design stage Graff & van Wyk [1, p. 97] Architectural diagram. Know where data is processed. User story, could draw one user story by time at board. Annotate what data is being transferred, where it crosses borders. Where are configuration files.

Many cases you can’t draw the physical machine. When you have all the data flows, when diagram is ready for now. Use lists like MS SRIDE , go through one data flow at time and write down the results. They may come in various form like testing needs or need of redesign of the architecture. When there is no time there might appear ignoring, rejecting risks.

A picture is a thousand words

A graphical presentation of analysis reveals details of a system. Consider also psychological aspect, memory of human is limited. Picture shows connections, might reveal something which is otherwise left unnoticed.

The most useful things to draw are ”Data Flow Diagram” and ”Message Sequence Chart”. The Data Flow Diagram shows processing blocks and data flows between them. And ”Message Sequence Chart” shows message traffic between communicating entities, related to time. The latter is Useful especially for more complex protocols (complex = more than two parties, or more than two messages). One way to work is to draw a Data Flow Diagram.
and for the most interesting (security point of view) data flows, draw a Message Sequence Chart. Analysis is against a running instance of a system, so static diagrams like class is not so useful. Use cases are also better get in plain text, than a diagram. Diagrams need to be detailed enough, compare drawings at figure 2.

You can stop at the first processing block that you do not control (develop), and leave that as a black box. For example, if you have a pure server side web application, the browser is a black box. (If you have client-side JavaScript, then it is not black box, you need to model it.) The data flow to/from that block still needs to be illustrated out in detail. All data flows in your Data Flow Diagram must have endpoints, they cannot end in the air, rather than need a box to end to.

Sometimes a data flow stops at a human being, these should be drawn. Your attacker is human. Humans to be drawn include users and admins. Remember that any human could be an attacker, so a source of attacker-controlled data.

Include also in a diagram admin interfaces and don’t forget to model key storage (including SSH keys, etc.), configuration files, load balancers (even if they would terminate TLS!) or client-side code execution (e.g. JavaScript library) in web applications. Note that in many cases you can’t draw the physical machine.

Drawing tools are not that good, better draw on whiteboard or on paper. Drawings are useful to do together with a team on whiteboard, there are pairs of eyes present at same time, you can get comments right away. Most useful thing to draw Data Flow Diagram, Message Sequence chart. Model
what you control.

Understanding Security Boundaries

Border of boxes, seen also in figures 2, in your Data Flow Diagram are security boundaries. When there is something you don’t trust, need to split the box. Within security boundaries of a physical machine. Containers (e.g., created using Mandatory Access Control) There are many types of containers that are usually run on top of an operating system. The Java VM is one example. Mandatory Access Control (MAC) frameworks such as AppArmor are other examples. Containers can be isolated execution environments, or they can just mediate accesses to resources. Example: Java VM and browsers JavaScript engines are execution environments. Example: AppArmor is an example of a system where the operating system checks whether a process has a right to do specific things; its not a separate execution environment but how the process interacts with other parts of the OS are checked and enforced. A process, including threads, they don’t have box of their own than shares one of the process.

A virtual machine is a security boundary Processes modern operating systems do not let processes to alter the execution of other processes A process can be inside a VM, which is inside a physical box...

Beware of fake boundaries Just deciding to store stuff in different directories is not a boundary, unless directory access controls are enforced from outside. Processes owned by the same user are really in the same domain (because a user can control all processes of that user)

An attacker that can execute code within a security boundary is usually thought of being able to control everything that happens within the boundary, and have access to all data within the boundary. Although this could sometimes be tricky, the safe assumption is to assume this. Example: If an attacker can execute code within a kernel, the attacker is thought to completely control the operating system - including data flows through it. Example: If an attacker can execute code on a physical host, the attacker is thought to completely control everything in all VMs running on that host.

Taint Analysis

In addition to possible input tainting, also communication protocols can get tainted. A protocol stack is layers of communication protocols, data flows through all the layers, at sending side from top to bottom, at receiving
receiving from bottom to top. For example: a typical web page load, where IP is responsible for getting its payload routed through the Internet. TCP (from port to port) creates a stream of octets by combining several IP packets, TLS on top of TCP provides confidentiality and integrity protection services, HTTP on top of TLS provides the web page request and response, including the web page itself, and the web application may then have an application level protocol that uses HTTP to exchange, for example, JSON objects.

```
client
Application
---------
JSON
---------
HTTP
---------
TLS
---------
TCP
--------
IP

server
Application
---------
JSON
---------
HTTP
---------
TLS
---------
TCP
--------
IP
```

The termination point 3 is part of your ATTACK SURFACE. You could think of it as a point of code that gets exposed to external inputs.

### 0.1 Taint analysis and security boundaries

Taint analysis and security boundaries, in which boundary does tainted data end up in. Identifying robustness test needs based on taint analysis.

Prioritising findings It does matter where the data came from. If you can trust the sender of the data, and can authenticate the sender, you might be able to decide that this potential way to inject malicious data is not a risk. (If you actually authenticate only trusted senders, that is.) Technically, everything that is possible may not be a security risk. EXAMPLE: Lets assume you have a Bluetooth-equipped system and you have identified two attack surfaces: An endpoint that receives data objects over Bluetooth from anyone An endpoint that receives audio data from a paired headset then it is most likely that the former one is a bigger risk, and the latter one isnt, because in that case, only a paired (trusted) party can talk to you. You will then want to prioritise the former. Of course, the attack surface that is
actually the Bluetooth pairing protocol is another question - if that can be subverted, then the second one becomes higher priority again.

Security Services for Data Flows & Stores; STRIDE

After discovering the data flows and data stores, we will look into security services we need to provide for each secure box and communication between them. Old model of security is CIA from Confidentiality, Integrity and Availability. New thread moelling acronym is SRIDE from Microsoft, http://www.microsoft.com/security/sdl/adopt/eop.aspx. Acronym comes from from words Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege.

Spoofing | pretending to be somebody else  
Tampering | Data is not tampered  
Repudiation | Usually non repudiation in banking, you do have logs. Repudiate vs non repudiation  
Information Disclosure | (TLS proprely)  
Denial of Service | E.g. flooding of sevie requests.
Elevation of Privilege | something is executed on higher level than given.

STRIDE maps to ”CIA triad” Inspect one data flow at time, write down results. Results come various in forms like testing needs or re-arhitecting.

Risk Analysis

You need to decide whether the cost of mitigation (fixing the issue) is greater than the risk (likelihood of it happening, and its impact). How this process is ultimately driven depends on your organisation and whether you have some sort of mandatory risk management process. However, from experience, here is some practical advice to you as a software engineer. All findings need to be ranked, and the actions are according to given rank and class of finding. The Risk is impact multiplied likelihood. A typical thing to calculate is an annual loss expectancy, that is, how many times per year the issue is going to cost you, times how much it is going to cost you each time.

References

