Threat Modeling

What, Why & How

IriusRisk

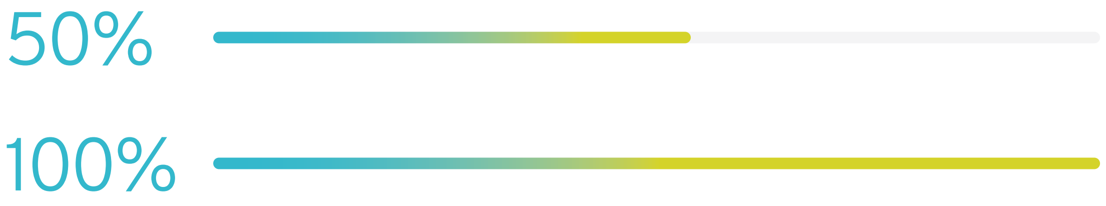
« What is Threat Modeling?

Threat Modeling Fundamentals

The fundamental basis of threat modeling is identifying, communicating and managing security weaknesses. This is achieved by understanding potential threats and attacks the system must resist and the corresponding countermeasures (controls) for those threats.

The key principle underpinning threat modeling is “secure design” which means in practice addressing design will take place from the inception of the project at the design phase and continue throughout the development life cycle.

Fixing in production vs Secure by design



Why should we threat model?

Cut Time to Production

Traditionally, security reviews have taken place towards the end of the development life cycle at the point in time in which it is the most expensive and time consuming to perform the analysis.

Research has found it takes 5 hours at the coding/development stage, testing phase and 10–15 times longer in production

This research also found that early and ongoing security testing cuts the time to production by as much as 25%.

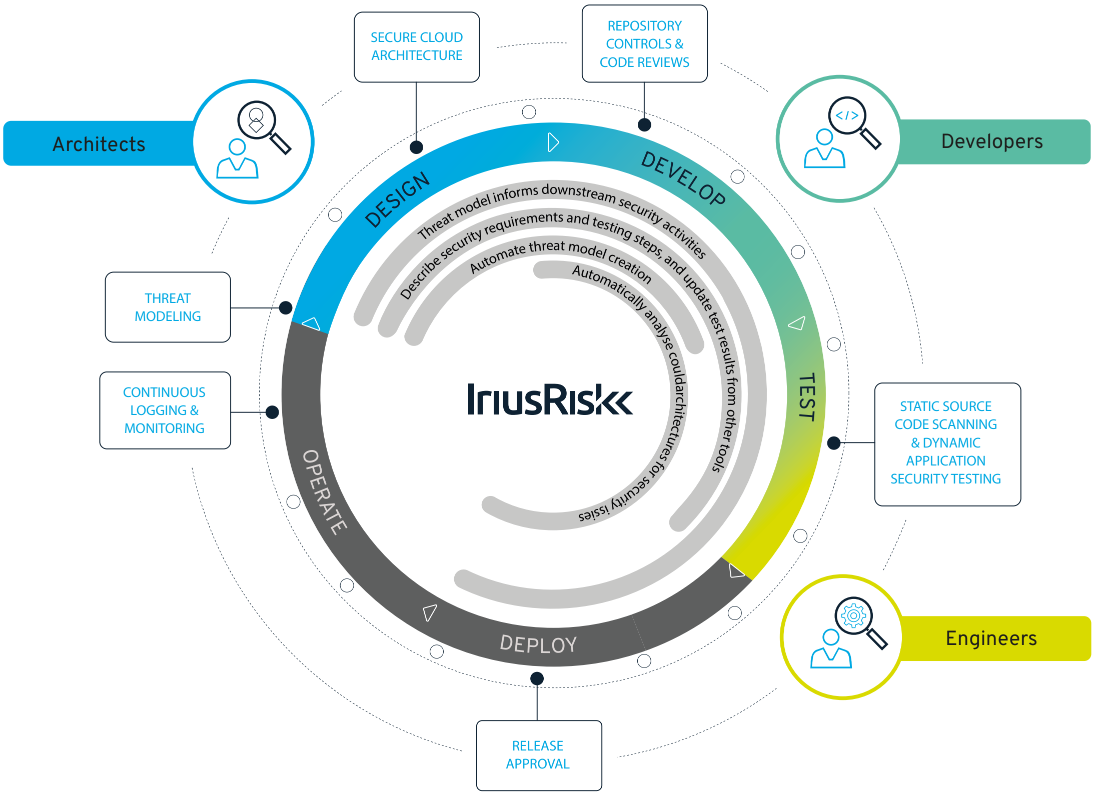
This type of research confirms identifying and mitigating security weaknesses at inception of the development life cycle and continuing throughout, is both cheaper and easier, which in turn is one argument for threat modeling activities having one of the best ROI within a security context.

Formulate Your Design

Threat modeling should begin when the functions and components of a product are known and the architectural design is being formulated. In this way threat modeling drives security requirements from the outset and has the dual method. This also ensures building and weaving security into the process rather than bolting it on at the end.

Threat Model at any Design Stage

It is worth noting here that if for whatever reason threat modeling cannot be integrated in the early design stage, benefits from this activity may be derived at any stage (even in production) of the product at hand and for the next development life cycle as you gain an understanding of the risk trade-offs that were made implicitly. Also, as many products have commonalities, elements of previous threat models may be reused and risk patterns and templates formulated.



SECURE CLOUD ARCHITECTURE
REPOSITORY CONTROLS & CODE REVIEWS
Architects
Developers
DESIGN
DEVELOP
TEST
DEPLOY
OPERATE
Threat model informs downstream security activities
Describe security requirements and testing steps, and update test results from other tools
Automate threat model creation
Automatically analyse couldarchitectures for security issies
IriusRisk
THREAT MODELING
CONTINUOUS LOGGING & MONITORING
STATIC SOURCE CODE SCANNING & DYNAMIC APPLICATION SECURITY TESTING
Engineers
RELEASE APPROVAL

How do we threat model?

The Threat Modeling Process

The process of threat modeling usually starts with a diagram which describes and maps the architecture, components, trust zones and authentication flows. It is very useful to incorporate data flows within the diagram:

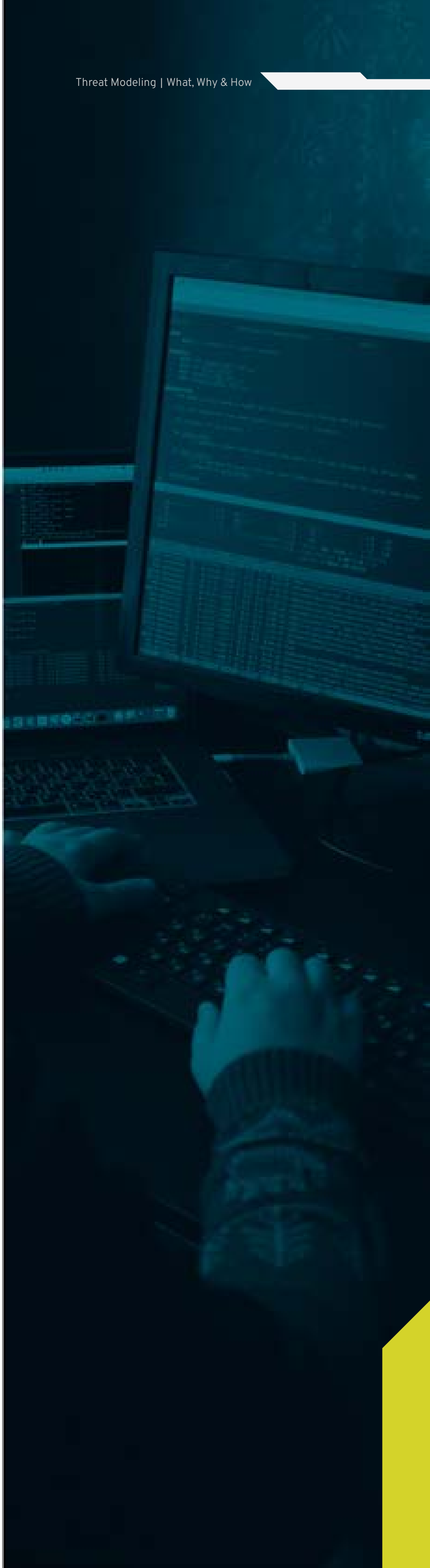
Data flows show how information enters and leaves the system, what changes the information and where information is stored. The purpose of a data flow diagram (DFD) is to show the scope and boundaries of a system as a whole.

Methodologies

Within the diagram potential weaknesses and attack points are identified. This threat enumeration activity may be promoted and facilitated by referring to attack trees (MITRE or CAPEC) or standards (PCI DSS, GDPR, SO, NIST) or guides such as OWASP's Mobile Application (MASVS). Also the Common Weakness Enumeration (CWE) and Common Vulnerabilities and Exposures (CVE) are useful here.

Microsoft's STRIDE Methodology is a popular tool for thinking about security threats within six categories; namely: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege.

*Learn more about Methodologies **here**.*



Collaboration is Key

This process may be formal or informal; performed on a whiteboard or through software, but two factors are paramount. Firstly, the diagram must be an accurate representation of the system, and secondly, as many as possible of the cross-discipline stakeholder partners involved in the project attend, the better. In this way a channel of communication is opened between security and other departments, giving room for assumptions to be challenged and differing viewpoints to be taken onboard.

Adam Shostack's Four Questions

At the heart of threat modeling is collaboration and communication with cross-pollination of ideas and knowledge between those with a range of expertise. Working together as cross-discipline teams will encourage a lingua franca to develop with enormous benefits for intercompany communication.

There are many threat modeling methodologies, but most can be encapsulated through four key questions articulated by Adam Shostack:

What are we building?

Architecture/ Data flow/
Data classification

What can go wrong?

Main threats that apply to your
application

What are we going to do about that?

Actions to take in response to this

Did we do a good enough job?

Retrospective analysis of the above

Benefits Seen Across Teams and Disciplines

As we can see, after the process of identifying potential threats and weaknesses, we must also consider the corresponding controls and tests. In terms of testing controls, threat modeling brings us an additional benefit in that it can be used to inform pentesters, auditors & reviewers of the: assets in play, controls already in place, boundaries and accepted risks.

Threat modeling is a team sport designed to elicit “tribal knowledge” and as such it is advantageous for security teams who are currently outnumbered 1.6 AppSec professionals per 100 developers to train up “security champions” who can act as coaches for their respective teams. This helps in spreading security knowledge and knowhow far and wide.

Threat modeling in an agile world

Integrating Devs and Ops

With the speed of development increasing in part due to the DevOps movement unifying software development (Dev) and software operations (Ops) together, and the corresponding move towards Agile development (with the emphasis on “sprints”), it is becoming progressively untenable to undertake the manual form of threat modeling.

As DevOps and Agile embrace continuous integration of automated tools along with potentially frequent architectural changes, it is imperative that we develop threat modeling tools in line with these needs.

Many of the threat modeling activities identified above can be automated. It is important that any automated threat modeling tool produces security findings and recommended countermeasures automatically, while preferably integrating with developers tools such as issue trackers so they are actionable and trackable.

This has the dual benefit of putting security knowledge into the hands of those building the product and in such a way that “speaks their language” via their own tools.

Ideally this threat modeling tool would come complete with threat libraries and regulatory standards templates built in, but also be extensible with the ability to add specific threats and standards relevant to that particular product or industry.

Other prerequisites for the ideal automated threat modeling tool to fit comfortably within a modern development environment would be:

Easy to use by **development teams**

Risk levels & mitigation implementation status identifiable and trackable

Ability to **integrate** with standard security testing tools

Granular control of user capabilities over the threat model

Bi-directional with developer issue trackers with each updating the other

Scalable as businesses adapt and grow

Ideal for **collaborative** work as a "Living Document"

Produces a visual **architectural diagram** with data flows

A **single point** to manage security throughout the development process