Security Testing

Better Testing with a Hacker's Mentality…

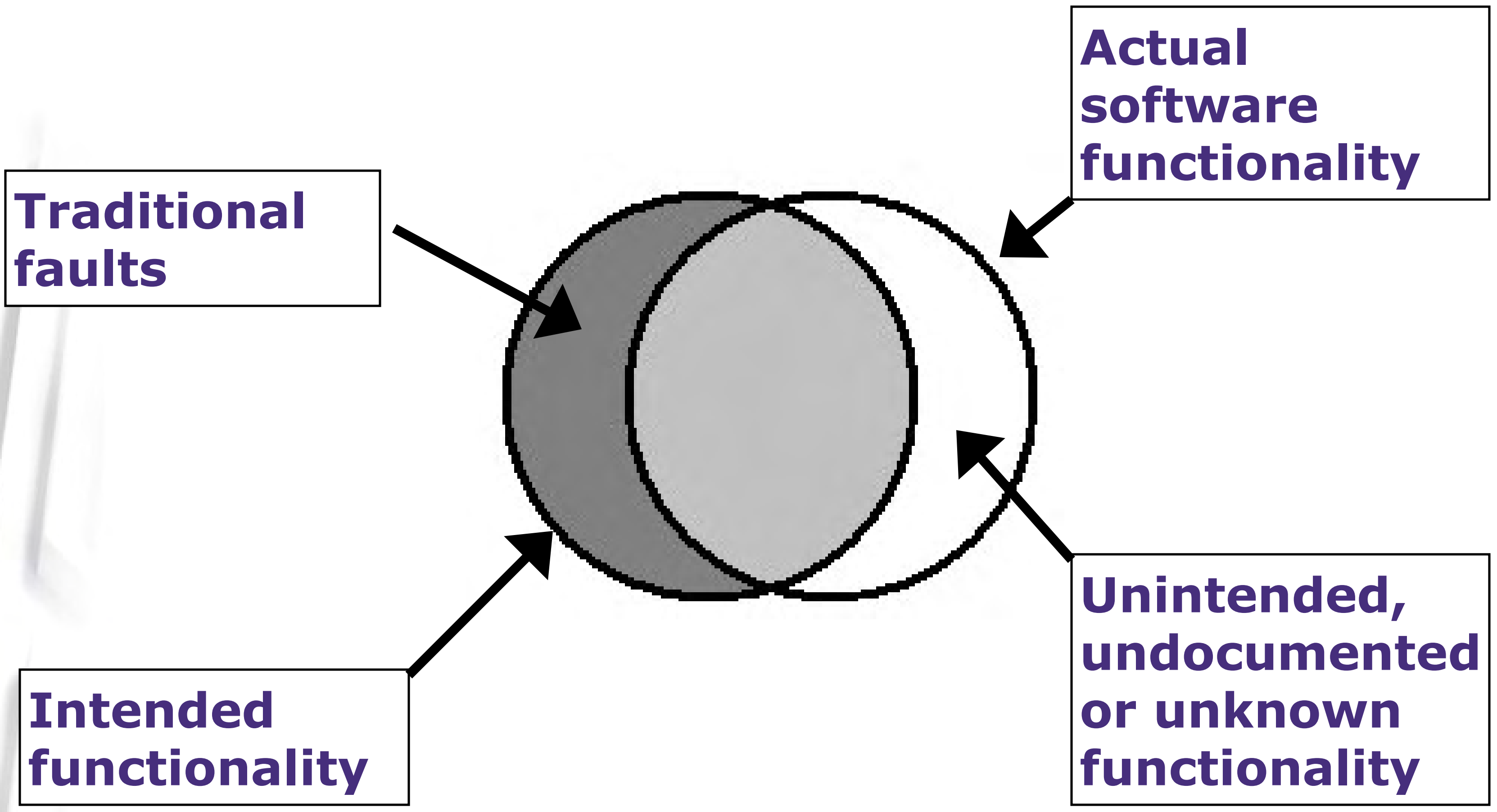
Julian Harty

Commercetest.com

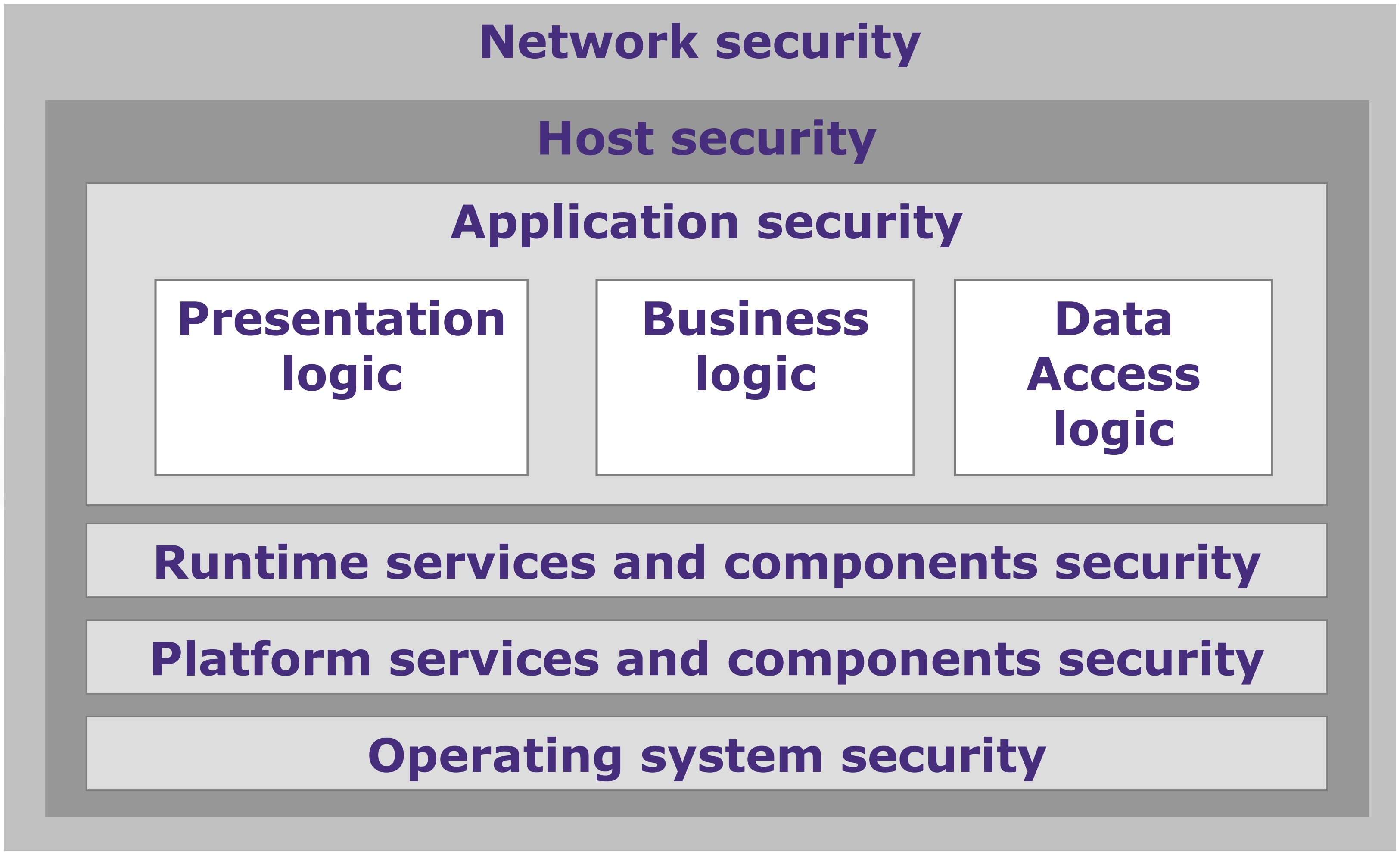
Scope of this presentation

- An introductory session, to get you started with security testing
- What is the “Hacker’s mindset” and how can we usefully apply it to find security problems before they do?
- Some practical, useful techniques
- How to get started in a safe environment

What's different about security testing?

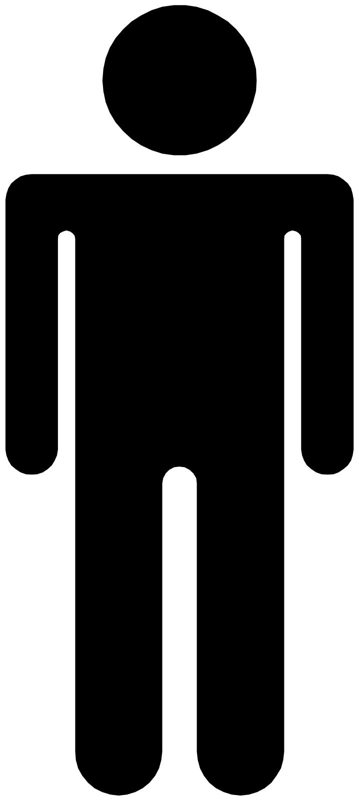

Traditional faults
Actual software functionality
Intended functionality
Unintended, undocumented or unknown functionality

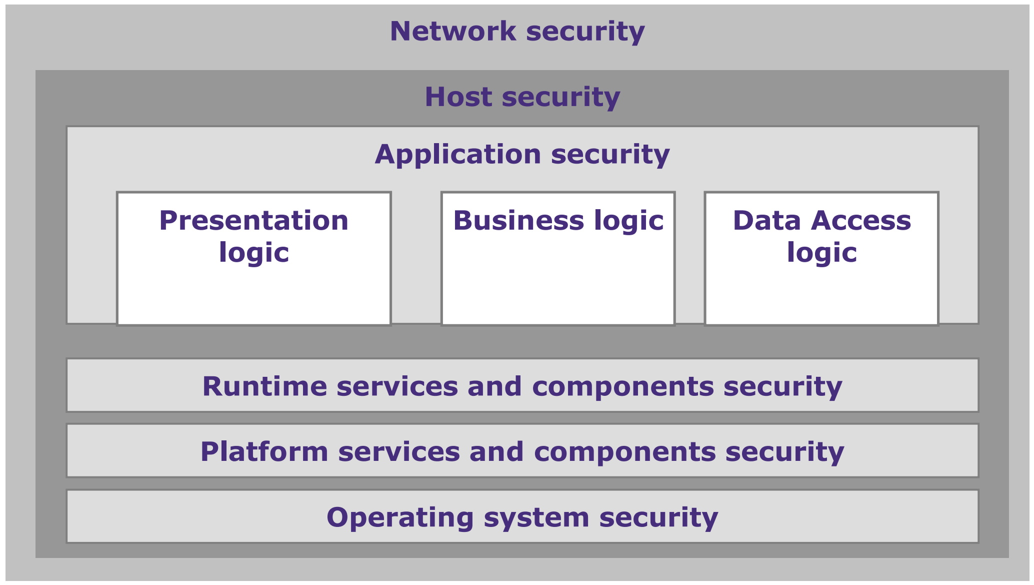
A holistic approach to security testing


Network security
Host security
Application security
Presentation logic
Business logic
Data Access logic
Runtime services and components security
Platform services and components security
Operating system security

Modelled on Microsoft security whitepaper - 2003

A holistic approach to security testing (2)




Network security
Host security
Application security
Presentation logic
Business logic
Data Access logic
Runtime services and components security
Platform services and components security
Operating system security

Defending against Layer 8 - **People**

Security ‘in depth’

- ‘Assumes no one item can deliver 100% security
- Layers of security are combined to provide better protection i.e. to make systems more secure

<u>What it means for testers</u>

- Testers need to test ‘a layer at a time’ as well as testing the whole
  - Testing becomes easier, and faults become clearer to attribute to the correct source
  - A key objective is to provide results that enable the system engineers to assemble the layers to provide the most effective & complete security

Threats, Vulnerabilities and Controls

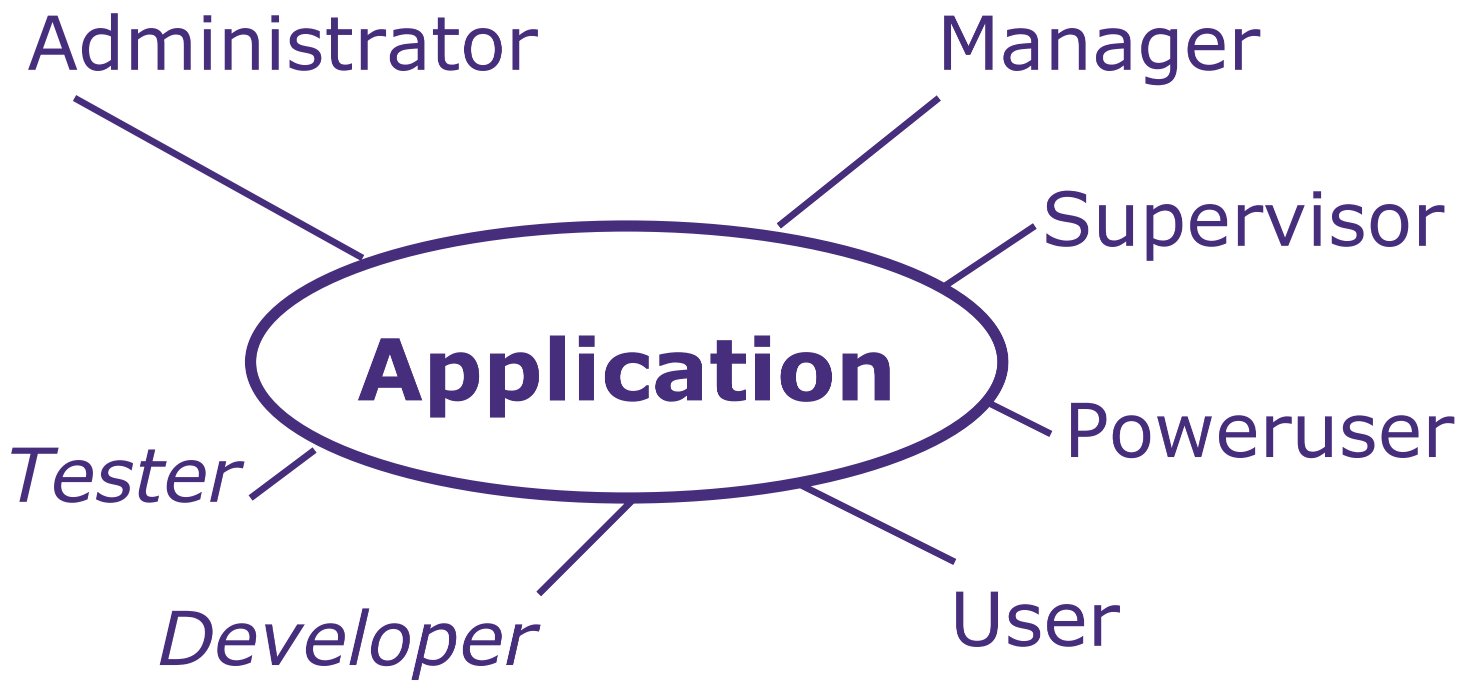
- **Threat** a set of circumstances that has the potential to cause loss or harm
- **Vulnerability** a weakness that might be exploited to cause loss or harm
- **Control** a protective measure that removes or reduces a vulnerability

- A **threat** is blocked by **control** of a **vulnerability**

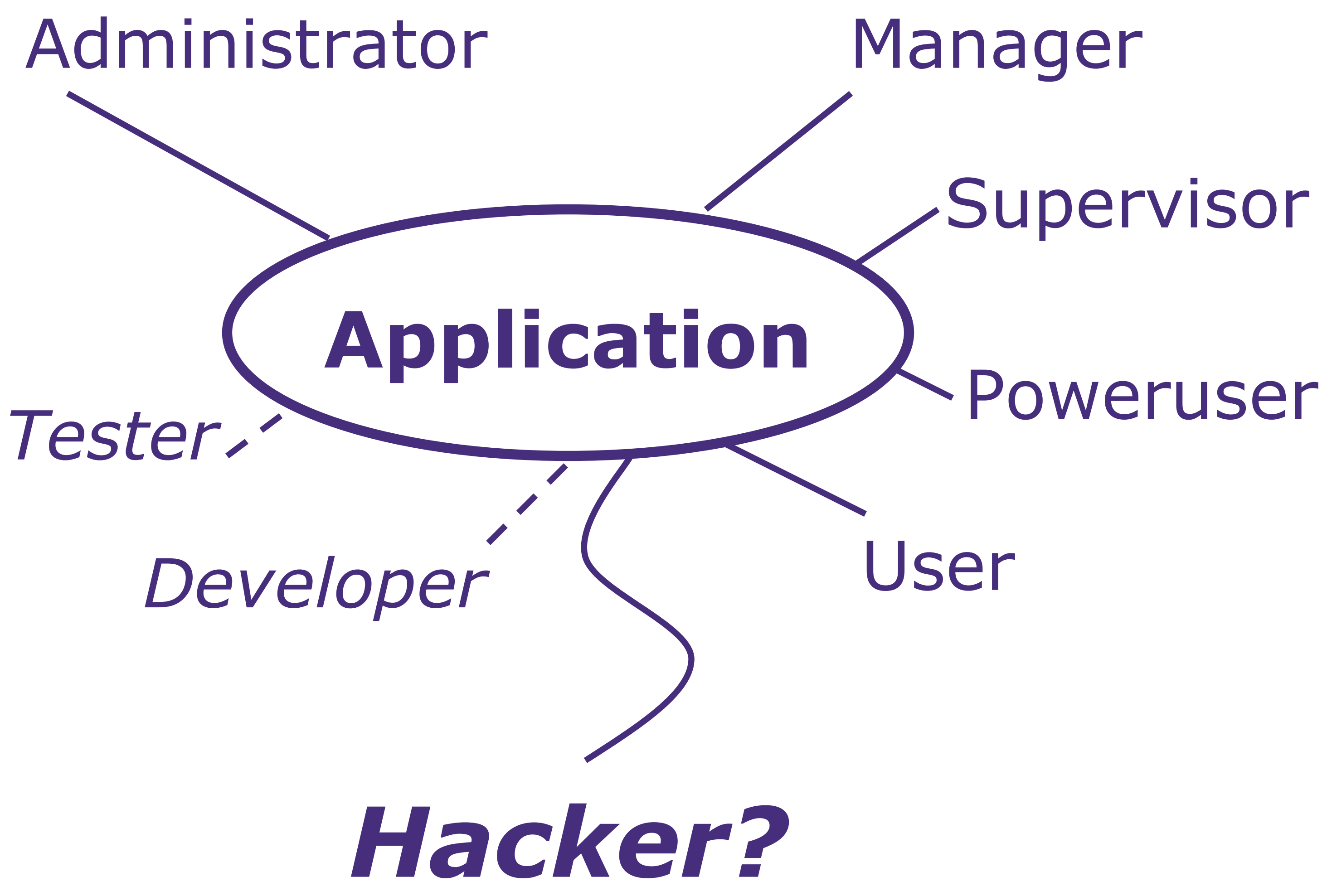
- Causes of Vulnerabilities:
  - Requirements
  - Construction
  - Operation

Definitions based on [Pfle2003]

Users of an application



Is the hacker a user of the system?



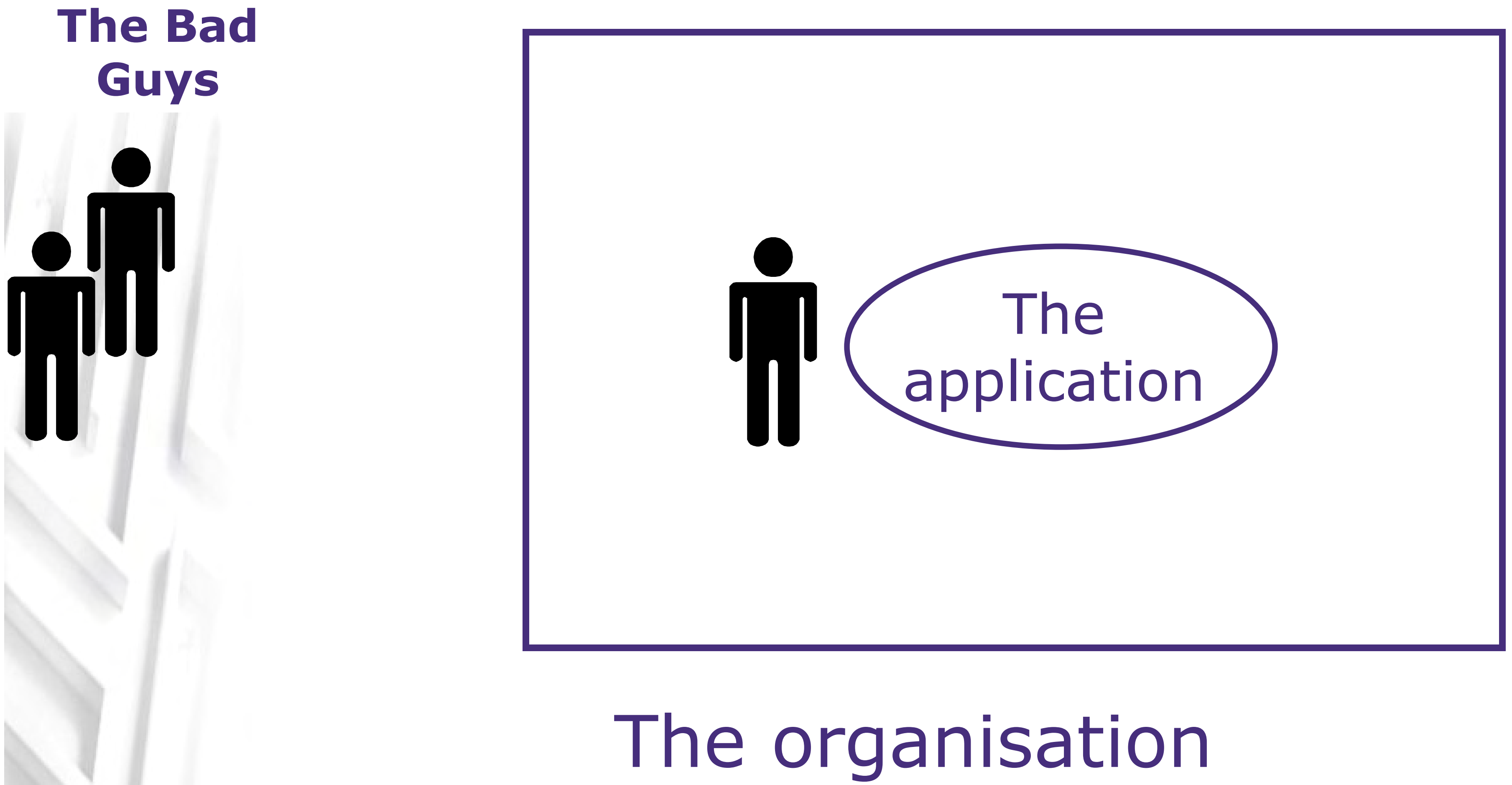
What is a Hacker?

- Someone who uses their technical skills…
- Through unexpected routes…
- To achieve their ***goals*** via the software application.

What is an Attacker?

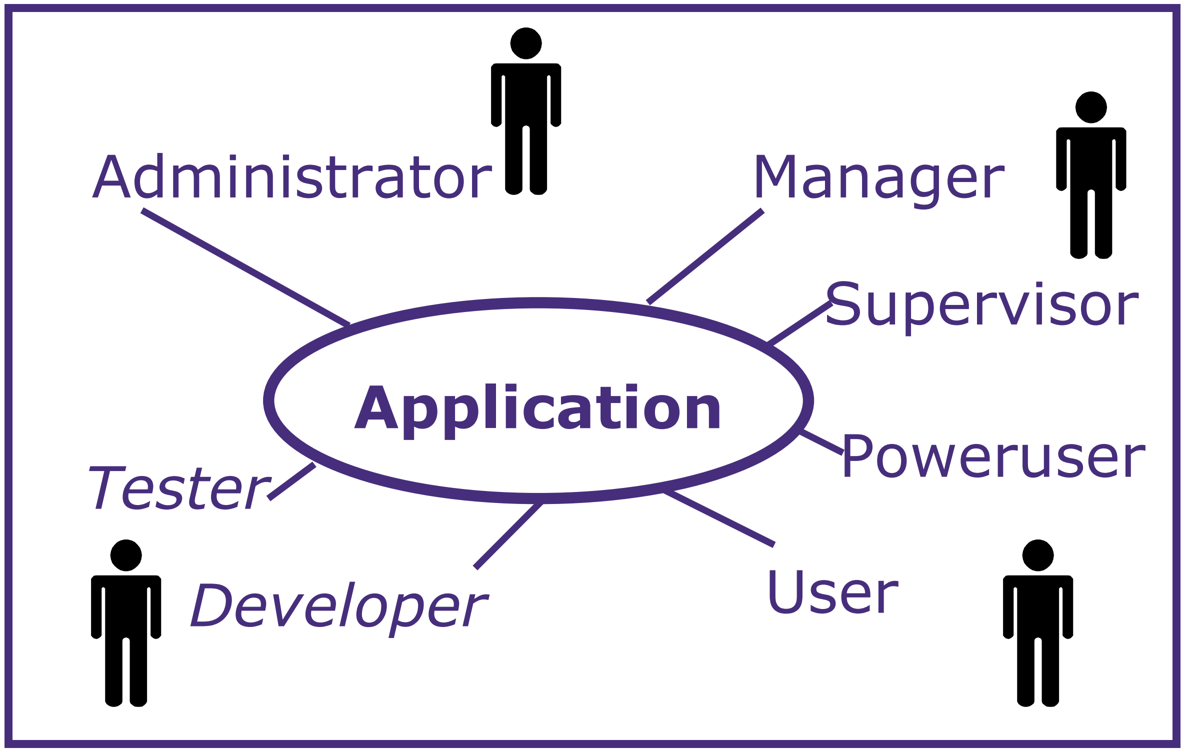
- Someone who uses their technical skills…
- Through unexpected routes…
- To achieve their ***nefarious goals*** via the software application.

Where are the hackers?



Who are the hackers?

The Bad Guys



The organisation

The Hacker's mindset

Characteristics

- Believing there **is** a way in! (persistence)
- No more "Mr nice guy" (forthright)
- Builds on the work of others (pragmatic)
- *Independent / Self-reliant*

Key drivers and success factors

- Competitive
- Cautious
- Skilled
- Clear targets

Persistence

Software is vulnerable, and will have security bugs that can be exploited.

As security testers, believe that you will find security bugs.

- Look for clues, be observant
- Believe you will get in, be successful
- Practice on less secure systems in a safe environment

No more “Mr Nice guy!”

The very nature of hacking means the Hacker isn’t too concerned with social niceties, and isn’t limited by notices to ‘keep out’.

As security testers, learn when to be forthright, and when it’s appropriate to transgress the conventional boundaries when testing the software.

However, unlike Hackers, our mission is to find and prove bugs, rather than exploit them for nefarious purposes, so plant flags, not bombs, when you reach your target.

Build on the work of others

There is a long history of one hacker building on the work / findings of another e.g. from

- Published exploits on bulletin boards
- Modifying and enhancing sample exploit code

As security testers, do the same:

- Read about techniques and exploits and try them out on your own system
- Review sample attack code to see how it works
- *BUT don't run **their code** on your system!*

Self-reliance

I believe (I'm a hacker) therefore I am!

As software testers, self-reliance, confidence in our skills, and an ability to work hard are all important. However we can take advantage of being one of the 'good guys' by:

- Working with other members of the team
- Establishing trusted networks with experts, peers, etc

So how *do* we gain the advantage?

- Small improvements can protect us from many attacks
- Use *insiders' knowledge* to find flaws before they do
- Use our existing testing skills to test more effectively
- We get a *head-start* so problems can be addressed before 'go live'
- Find allies in the war against attackers
- Leverage the work of software vendors to:
  - Ensure our systems are patched and protected
  - Learn from their mistakes and how they fix things
- Learn in a safe environment

CMP United Business Media

Wireless Wonders! Wi-Fi, ZigBee, & Wireless USB

#366 NOVEMBER 2004

Dr. Dobb's JOURNAL

SOFTWARE TOOLS FOR THE PROFESSIONAL PROGR

DEVELOPERS' CHOICE EDC AWARD 2004 Most Trusted Publication for Tools and Software Technologies Dr. Dobb's Journal

http://www.ddj.com

COMPUTER SECURITY

Avi Rubin
World's Leading Expert on E-Voting

Practical Port Knocking

Secure Web Forms

Telephony & the Parlay Spec

Binary XML

Web Services & Apache Axis

Distributed Compilation

The Virtual Instrument Software Architecture

Windows Forms Layout Managers

Silent Application Update

Creating Trace Listeners in .NET

Jerry Pournelle On Digital Rights Management

Bonus Eclipse Coverage!

Eclipse & Tools for Embedded Systems

Factoring for Eclipse

$4.95US $6.95CAN

0 74470 01051 7 11>

XML Compression

Wik3 Book

November 2004

w.commercetest.com

Case study: Electronic Voting System

- Electronic Voting Systems are being used in a number of countries, and are claimed to offer unrivalled speed, accuracy and reliability.
- However there have been many reported concerns and complaints the process isn't as trustworthy as first claimed.
- Would the following techniques help to identify and test for possible security vulnerabilities?
- Consider a scenario where you are responsible for security testing of the product for your local state, or region – what would you want to test, and how?

Goals and Agents

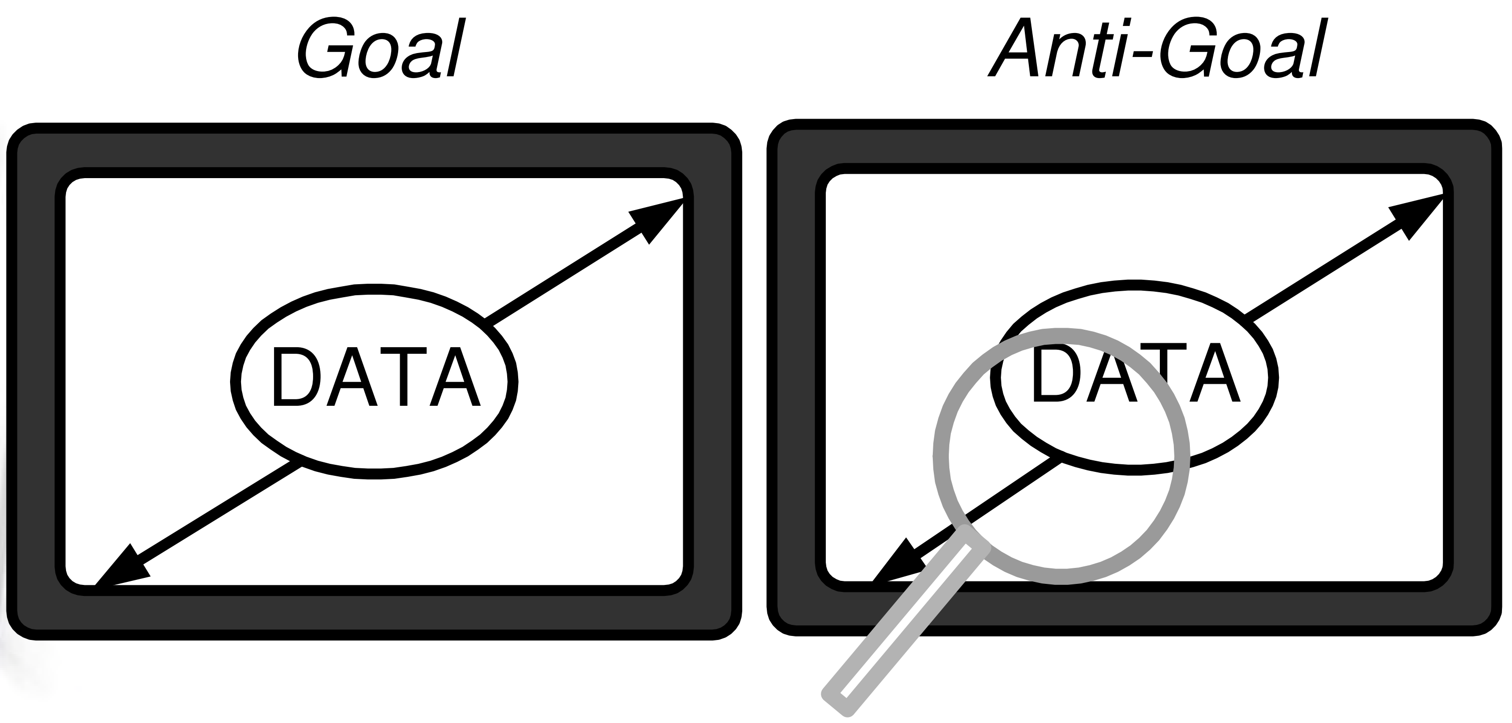
- **Agent** – active components e.g. humans, devices, software, that play some role towards satisfying a goal.
- **Goal** – a descriptive statement of intent about a system (existing or to-be) who's satisfaction requires the cooperation of some of the agents forming that system.

- Some Agents relate to the software, others to the environment
- Goals may relate to the services to be provided (functional goals), or to the quality of service (non-functional goals)

Anti-Goals – an overview

- Use the perspective of an Attacker and look at goals the hacker might want to achieve e.g.
  - Obtain login and password
  - Intercept sensitive e-mail
- Can model Attackers and their Agents, and their capabilities in terms of:
  - Operations they can perform
  - Objects they can monitor/control
- Can model software vulnerabilities
- Can model unlikely, but critical outcomes e.g.
  - Terrorist attacks

Anti-goal for confidentiality



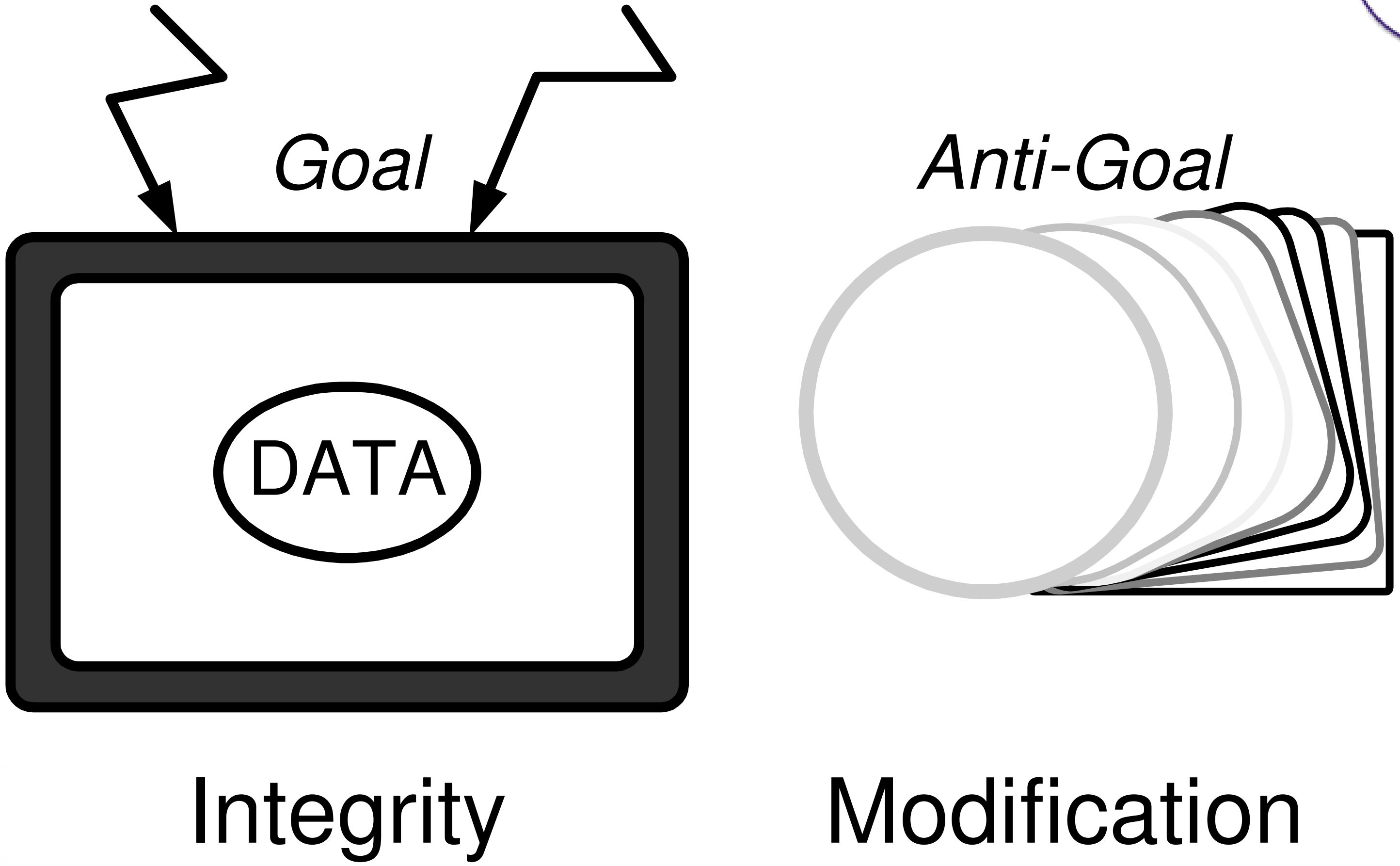
Confidentiality

Unauthorised disclosure

- Anti-Goals include Breaking-in, copying data, and may involve redirecting traffic through a ‘man-in-the-middle

Based on figures 1-2 [Pfle2003]

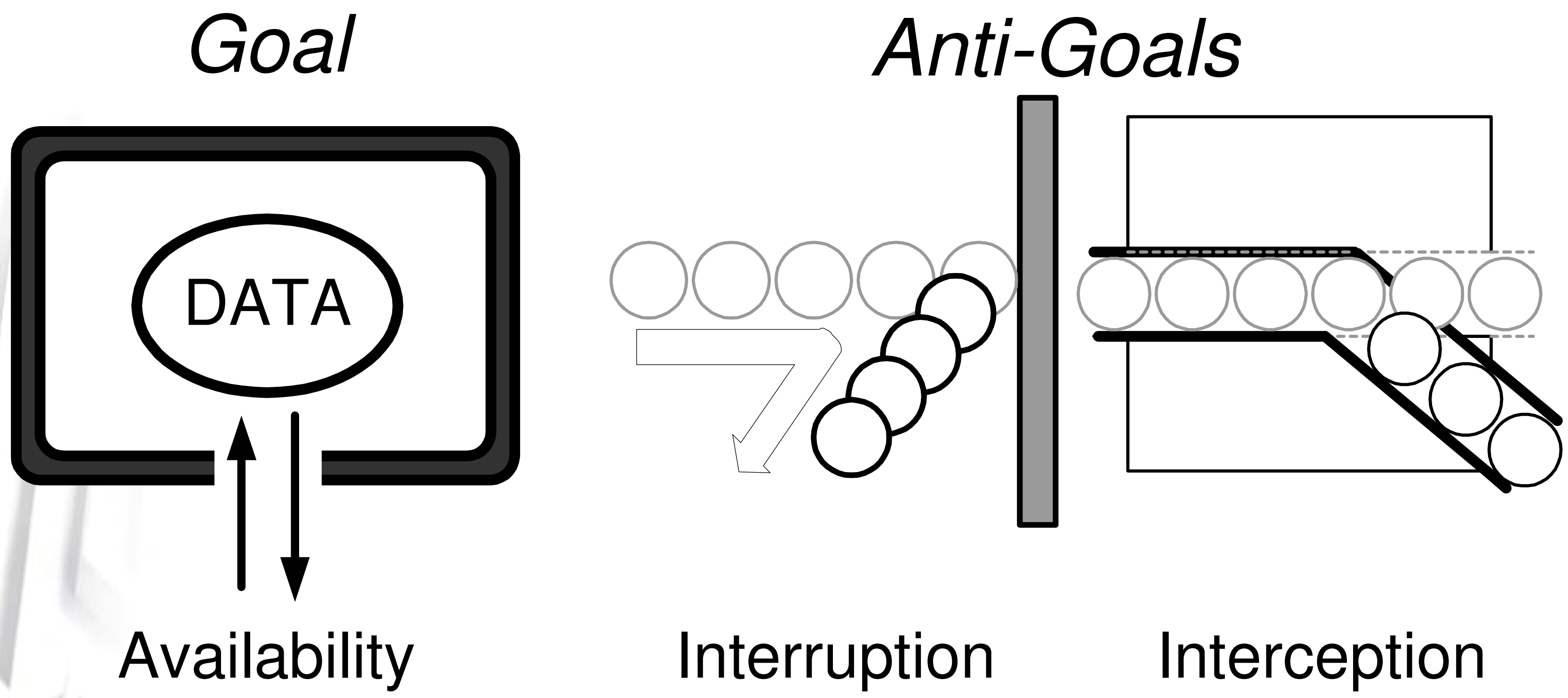
Anti-goal for integrity



- Modification is the anti-goal of integrity, as users can no longer ‘trust’ the system with their data

Based on figures 1-2 and 1-5 [Pfle2003]

Anti-goals for availability



- Availability can be attacked by interrupting or intercepting requests / responses

Based on figures 1-2 and 1-5 [Pfle2003]

Anti-goal: fabrication

- Attackers often fabricate, or fake, requests to try to fool the system into accepting their requests as if they were valid
- The attacker may want to obtain information (anti-goal of confidentiality) or to take control e.g. in order to modify data (challenging the system's integrity)

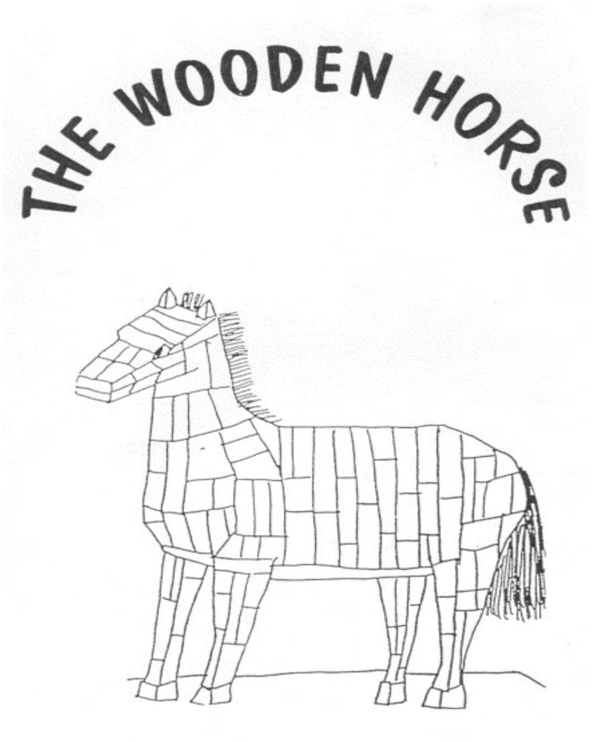
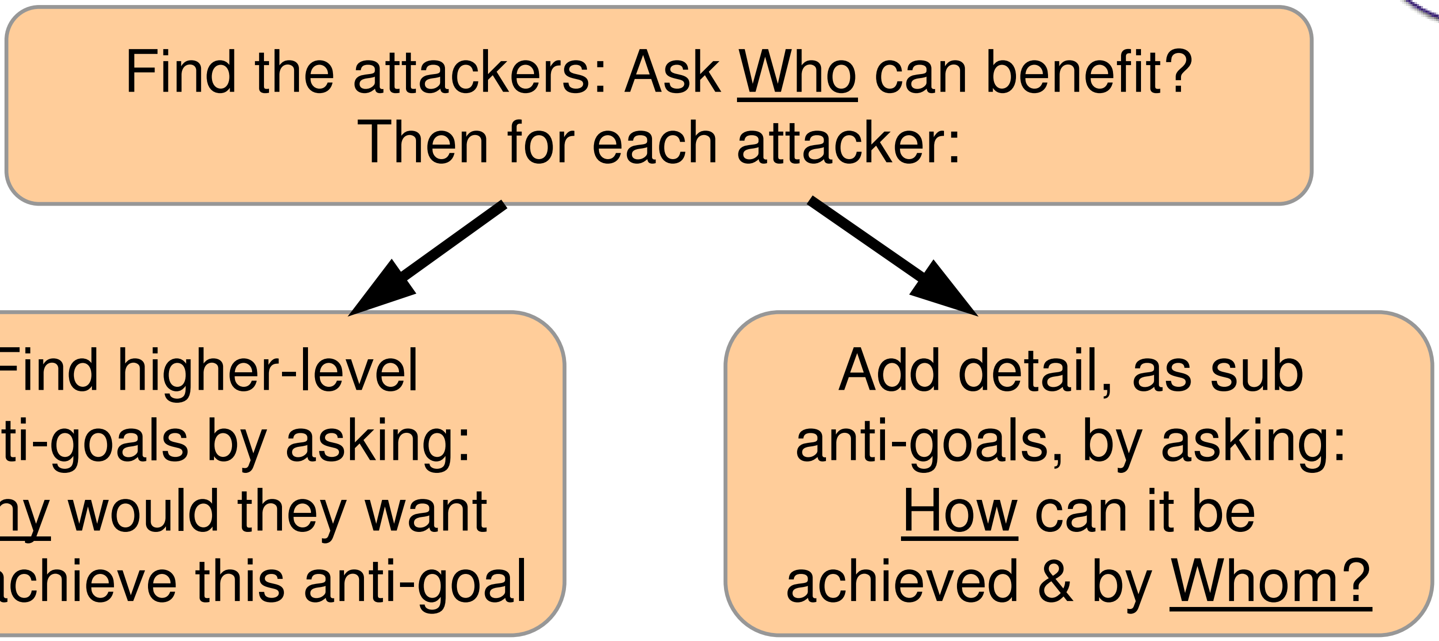


Image from: http://www.stcatherines.surrey.sch.uk/wooden_horse_of_troy_page.html

How-to identify the anti-goals of a system



Find the attackers: Ask <u>Who</u> can benefit?
Then for each attacker:

Find higher-level anti-goals by asking: <u>Why</u> would they want to achieve this anti-goal

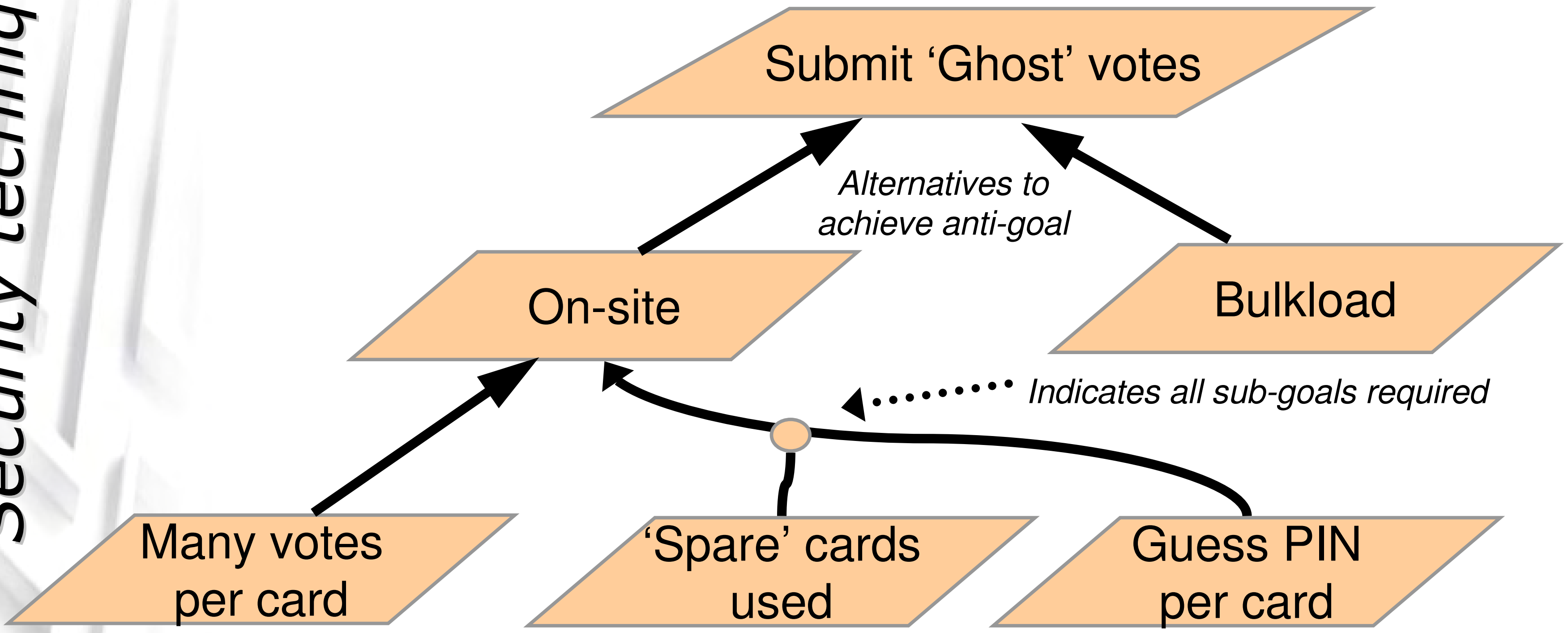
Add detail, as sub anti-goals, by asking: <u>How</u> can it be achieved & by <u>Whom?</u>

- If the anti-goal can be achieved by the attacker's agent, it's an <u>anti-requirement</u>
- Or it's part of our system and called a <u>vulnerability</u>
- When creating sub anti-goals, look for alternative ways to achieve the parent anti-goal, and identify those that need to be combined in order to achieve the parent anti-goal.

Create anti-goal model

Security techniques

- Create a diagram showing the hierarchy of anti-goals, and their relationships.



Part of an anti-goal model for Electronic Voting System

Anti-Goals – the process (1)

1. Negate security goals e.g. Integrity negates to Modification
2. For each anti-goals ask:
   - **Who** can benefit from this anti-goal?
3. Refine the goals and look for AND/OR relationships. Ask:
   - **Why** would this attacker benefit to find parent anti-goals?
   - **Do** I need more detail to make this anti-goal realisable?
   - (AND) Do I need to combine sub anti-goals to achieve the parent?
   - (OR) Is the sub anti-goal sufficient by itself to achieve the parent?
4. Refinement stops when terminal anti-goals are reached. These are either:
   - Anti-requirements if they're in the Hacker's domain
   - Vulnerabilities if they're in the System's domain

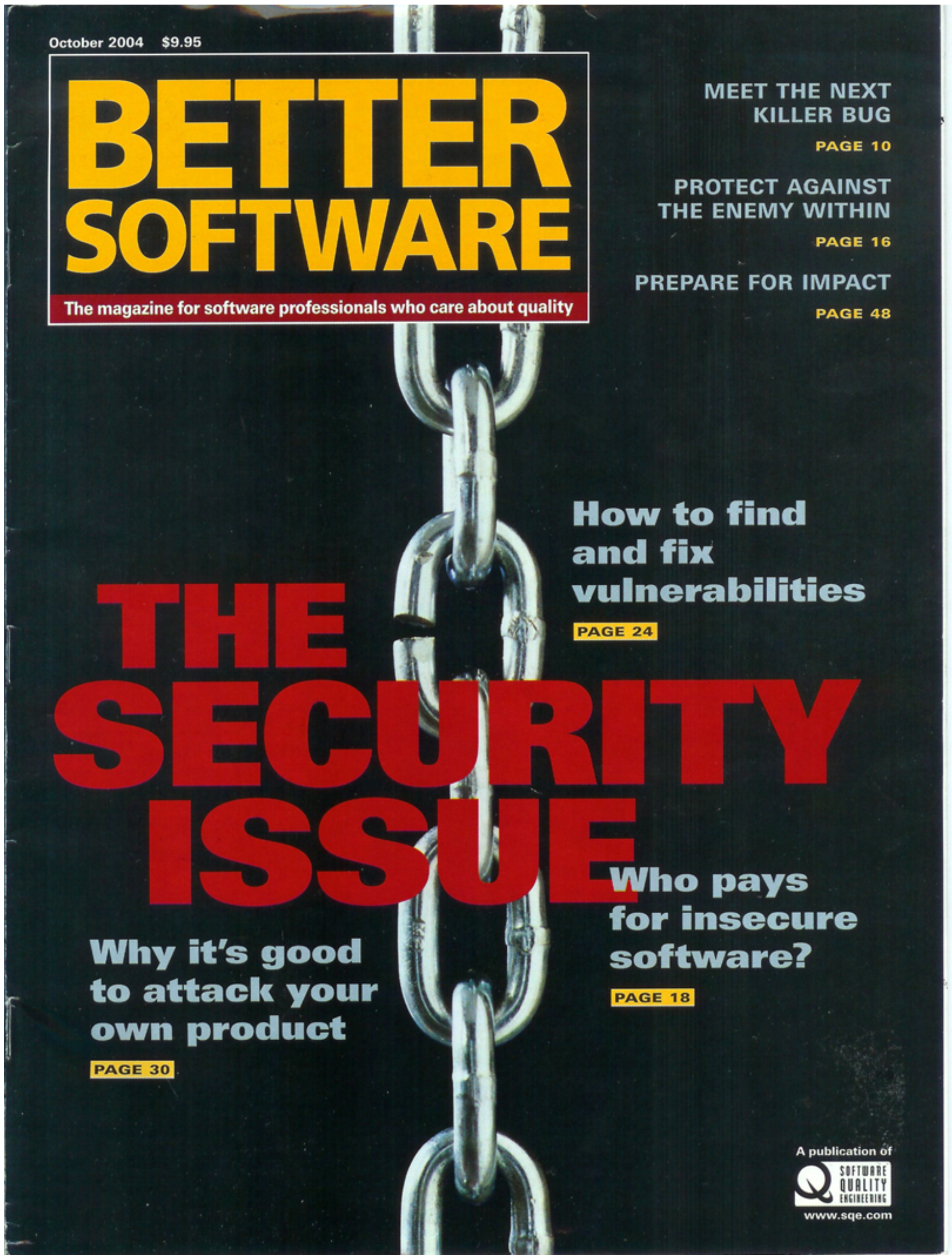
Anti-Goals – the process (2)

5. Find the boundary and monitoring/control interfaces between the:
   - What's under the attacker's control
     - (called the Anti-Machine)
   - Our software / system
     - (called the Anti-environment)
6. Identify the potential capabilities of the attacker for each anti-requirement e.g.
   - Eavesdropping
   - Deciphering
   - Spoofing
   - Blind/Intelligent searching

How do we use the anti-goals?

- As testers, we can use the anti-goals to create test cases, which we execute against the system being tested
- We can work with the analysts, designers and others to generate countermeasures. Countermeasures include:
  - Goal substitution
  - Agent substitution
  - Goal weakening
  - Goal restoration
  - Anti-goal mitigation
  - Anti-goal prevention

{ Protect vulnerability
Defuse threat
Prevent vulnerability



October 2004 $9.95

BETTER SOFTWARE

The magazine for software professionals who care about quality

MEET THE NEXT KILLER BUG PAGE 10

PROTECT AGAINST THE ENEMY WITHIN PAGE 16

PREPARE FOR IMPACT PAGE 48

How to find and fix vulnerabilities PAGE 24

THE SECURITY ISSUE

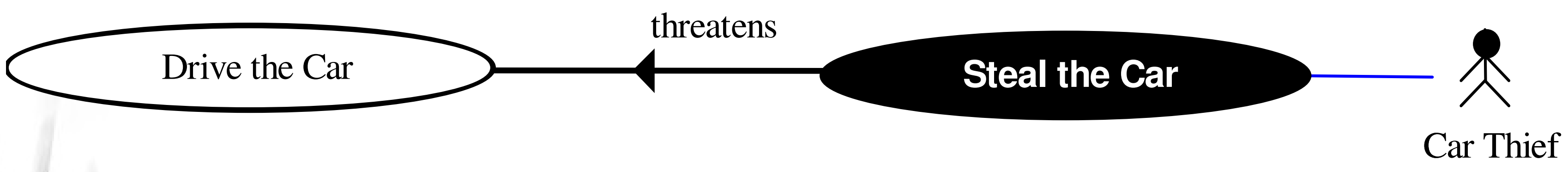
Who pays for insecure software? PAGE 18

Why it's good to attack your own product PAGE 30

A publication of SOFTWARE QUALITY ENGINEERING

www.sqe.com

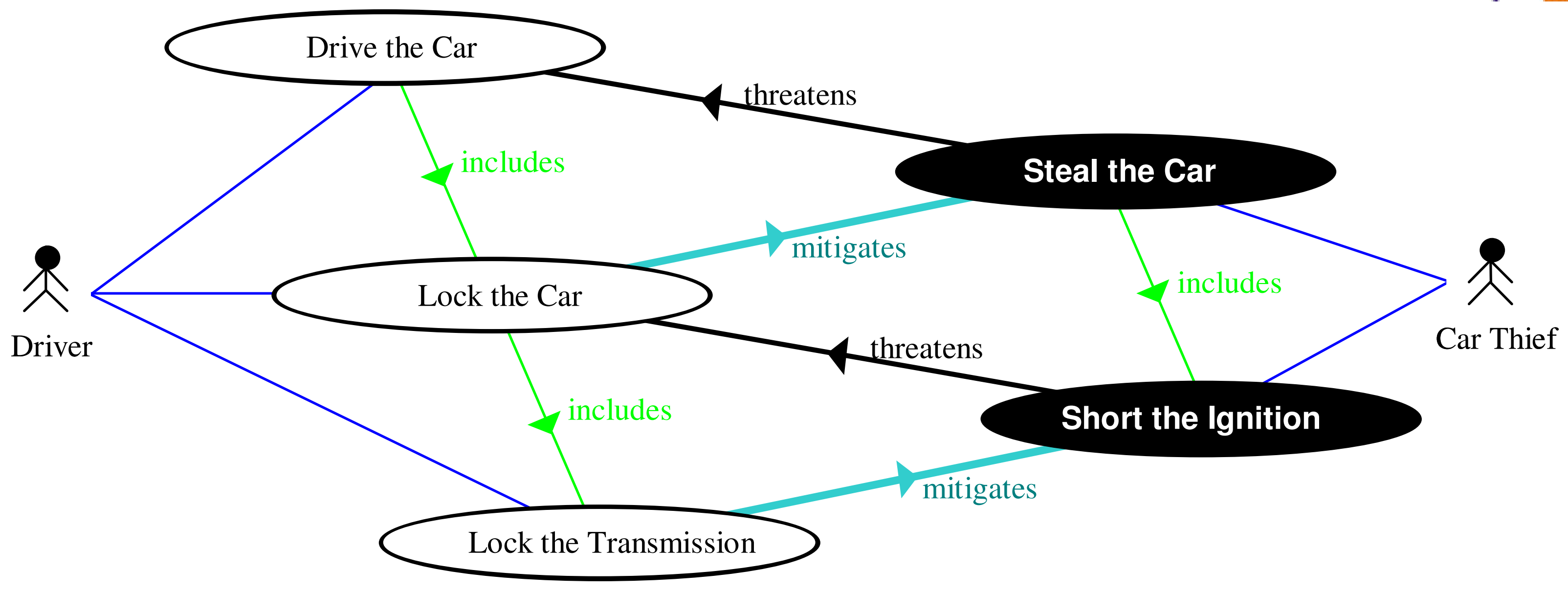
Misuse Cases



- Guttorm Sindre and Andreas Opdahl, 2000
- Enhanced by Ian Alexander, 2001..

- Actor is a hostile agent
- Bubble is drawn in inverted colours
- Goal is a threat to our system
- Obvious security applications, can also be used for other non-functional attributes

Misuse cases are related to Use cases



Use Cases for 'Car Security'

- UML's stick-man looks like 'human agent' but can be of any type (robot, system)
- White's Best Move is to find out Black's Best Move, and counter it
- A misuse case can 'threaten' a use case
- A use case can 'mitigate' a misuse case

Getting started - Be safe

Test environment(s)

- Isolate the test environment from *everyone else*
- Have a safe environment where *you can experiment and learn.*
  - Consider using the OWASP project

Protect yourself (CYA)

- You are working in a very sensitive area and have a responsibility to protect yourself from danger
- Neither be the only one who knows a fatal secret, nor the one who inadvertently tells the world…
- You share the responsibility for protecting the vulnerabilities you discover

Adapt techniques, material, and tools from the hacker's community

You do not know it all

- Be willing to learn about tools and techniques from the external community
- Adapt what hackers do to help you test more effectively, but aim to find problems, not cause them…

Tools

- Scripted penetration
- Automated testing
- Analysers
- Debuggers
- Environmental control

Techniques

- Buffer overruns
- XSS – attacking via client system
- Command injection
- Look for common mistakes ☺

Questions & Answers

Please get in touch if you'd like more information

julian.harty@commercetest.com

See also: www.testingstandards.co.uk

Security References

- www.commoncriteriaportal.org - Common Criteria material
- www.owasp.org - various tools and whitepapers (Java and .NET)
  - WebScarab: helps testers to find vulnerabilities & attack web sites
  - WebGoat: sample web application with inbuilt tutorials on common attacks
- www.microsoft.com - Threats_Countermeasures.pdf
- //easyweb.easynet.co.uk/~iany – Material on misuse cases, etc.
- www.testingstandards.co.uk - software standards and examples
- Security in Computing 3rd edition [Pfle2003]
  Charles P. Pfleeger, Shari Lawrence Pfleeger
  ISBN 0-13-035548-8, Prentice Hall © 2003
- How to break software security
  James A. Whittaker, Herbert H. Thompson
  ISBN 0-321-19433-0, Addison-Wesley © 2003
- See also:
  - An information security handbook
    John M. D. Hunter
    ISBN 1-85233-180-1, Springer © 2001
  - Testing Web Security
    Steve Splaine
    ISBN 0-471-23281-5, Wiley © 2002