Threat Modelling

Agenda

- Overview
- General infosec / network assessments techniques
- Attack Trees
- STRIDE/DREAD Threat Model approach
- Describing the Security Model for an Application
- Agile Security Model (asm)
    - Threat Modelling as part of agile software development
- Conclusions

Integrated Approach to Delivering Security

- Corporate security policy statement
- Security Operations (integrated with regular processes)
- Security Infrastructure
- Deploying penetration tools/analysers
- Security-aware users & I.T. admin / operations staff
- Security-aware developers
    - Occasionally supplemented by outside consultants; but need in-house expertise, preferably among all developers
- Application security design / code review

Security Policy

- "House rules" for how organisation deals with security
  - As much as possible, automate the translation from definition of security policy (xml) to actual implementation (scripting)
- Security is a team effort – everyone should know how to contribute
  - Security policy needs to be communicated to everyone (managers, end-users, developers)
- If security breach occurs:
  - I.T. manager/senior network admin/security officer look foolish if appropriate security policy is not defined
  - Developers, network operators & end-users look foolish if they have not followed a sensible security policy

Enterprise-Level Security Mgmt / Risk Assessment

- Use best practices/common sense for security of full information lifecycle
- ISO 17799 / BS 7799
  - Enterprise-level security management (e.g. used by WALMART)
  - http://www.iso17799software.com/
- Octave (Operationally Critical Threat, Asset, and Vuln Eval)
  - Self-assessment of own network – based on workshops
  - http://www.cert.org/octave/
- IAM (NSA's InfoSec Assessment Methodology)
  - 18 areas - InfoSec Doc, Roles and Responsibilities, Identification/Authentication, Account Management–Session Controls, External Connectivity–Telecommunications, Auditing–Virus Protection, Contingency Planning–Maintenance, Config Mgmt–Back-ups, Labeling–Media Sanitation/Disposal, Physical Environment–Personnel Security, Training & Awareness
  - http://www.iatrp.com

NIST 800-30

- NIST = U.S. Government's National Institute of Standards & Technology
- Special Publication 800-30 = “Risk Management Guide for Information Technology Systems”
  - http://csrc.nist.gov/publications/nistpubs/
- Practical discussion of risk management for IT applications, hosts and networks
  - Quite readable & low ceremony
- OWASP will shortly release threat model spec based on this – see www.owasp.org
  - Open Web Application Security Project

Multiple Layers of Defence

- Network Security
  - Discovering layout of network and vulnerability points
  - Three excellent books – (1) “Network Security Assessment”, McNab, ISBN: 0-596-00611-X (2) “Assessing Network Security”, Lam et al., ISBN: 0-7356-2033-4 (3) “Network Security Hacks”, Lockhart, 0-596-00643-8
- Host Security
  - Cracking OS and layered services (directory, file system)
  - User logins, privilege breaches
- Application Security
  - Main focus of this talk
  - App is primarily accessed via network protocols, data schemas (SQL, XML, binary), programmatic APIs, user interface

Goals

- A systematic approach is needed to discover, document and defend against the threats our software applications face
- Building attack trees, threat models and security models gives us a heightened understanding of the potentially hostile environment within which our applications must securely execute
- Threat models (and general secure development) must be sensibly integrated with our (preferably agile-based) software engineering process
    - This talk is aimed at experienced software engineers who know the fundamentals of security and are now interested in exploring how to design, code and test secure software

Is it needed at all?

- “Rather than spend time on threat modelling, shouldn't we do just all that is possible to improve our software”
  - “After all, most developers do not perform threat modelling”
  - Most developers have apps with security holes!
- Modern software applications tend to be large, complex beasts, managing assets of varying values, and facing more or less likely series of threats
  - Obviously we should get into habit of automatically preventing simple threats and implementing every no/low-cost measure
- There are always limits on resources (developer time), so we need to focus on what is important / likely
  - Credit card information on an eCommerce site is simply more important that press releases

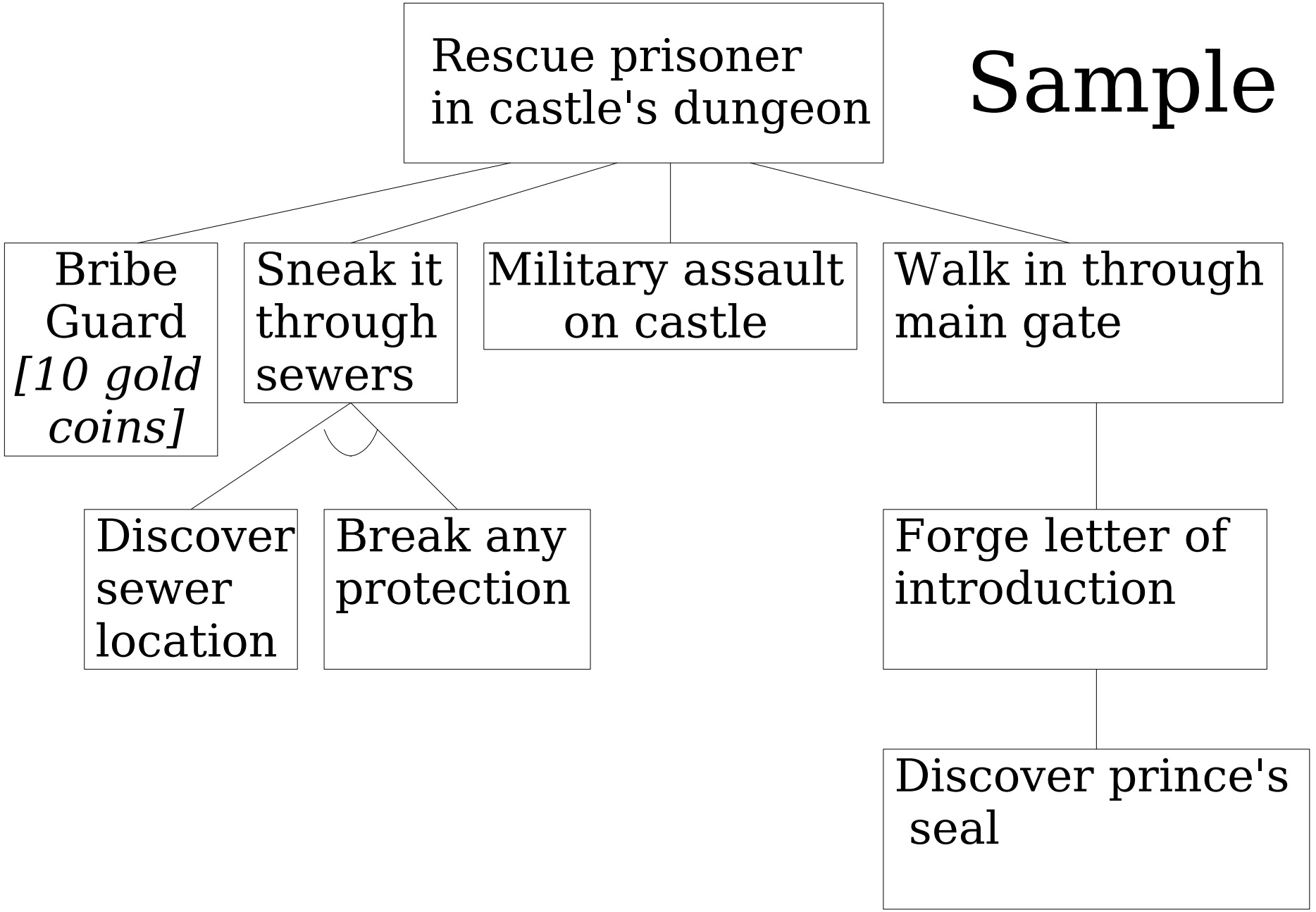
Minimising Attack Surfaces

- Attack surfaces = the possible routes over which attacks can flow
  - The more “visible” you are to the enemy on the battlefield, the more likely you will face attack
  - Includes: sockets, files, daemons/Windows Services, user accounts, access control settings (files), web handlers/filters
- Categories of attack surfaces
  - Any service, service as SYSTEM/root, service with low privilege
- By focusing your attention on a smaller number of attack surfaces, likely to be able to protect them better
  - One issue for security is the sheer scale of data – need to minimise that
  - Management by exception

Attack Trees

- Threats are often based on a combination of factors
  - A little weakness here combined with a little weakness there
  - Attack trees try to identify these combinations
- Consider target as the destination
  - Often multiple steps needed in some logical sequence
  - Often multiple routes can be travelled to reach it
  - Describe attacks as a tree of nodes (sub-trees may be shared among attack scenarios)
- Attack has to start out on that route
  - often the most difficult
  - Can be a long way from the destination (those who need to protect the destination must understand this)

Sample


Rescue prisoner in castle's dungeon
Bribe Guard *[10 gold coins]*
Sneak it through sewers
Military assault on castle
Walk in through main gate
Discover sewer location
Break any protection
Forge letter of introduction
Discover prince's seal

How to Travel

- Attackers can travel along attack route via:
  - Their own technical skills
    (e.g. expert knowledge of TCP packets)
  - “Interesting” information that is left lying around
    (e.g. passwords in trace files)
  - Cash (bribery)
  - Eavesdropping
  - Perseverance (running penetration tools)
  - Patience
    (intermittent carelessness on the part of authorised users)
  - Good luck
- Each node may have a value or condition
  -

STRIDE / DREAD Threat Modelling

- Risk assessment for applications

Threat Modelling Steps

- Identify Assets
  - What is of value
- Create an Architecture Overview
  - High-level architectural description (diagram, use scenarios, technologies)
- Decompose the Application
  - Identify entry points, exit points, trust boundaries, data flow descriptions, the privilege for various categories of code
- Identify the Threats
  - Think about how app can be attacked (STRIDE, vulnerability Databases, experience, attack trees)
- Document the Threats
- Rate the threats (DREAD)

STRIDE – Main Threats

- Spoofing
- Tampering
- Repudiation
- Information disclosure
- Denial of service
- Elevation of privilege

DREAD – Defining Risk

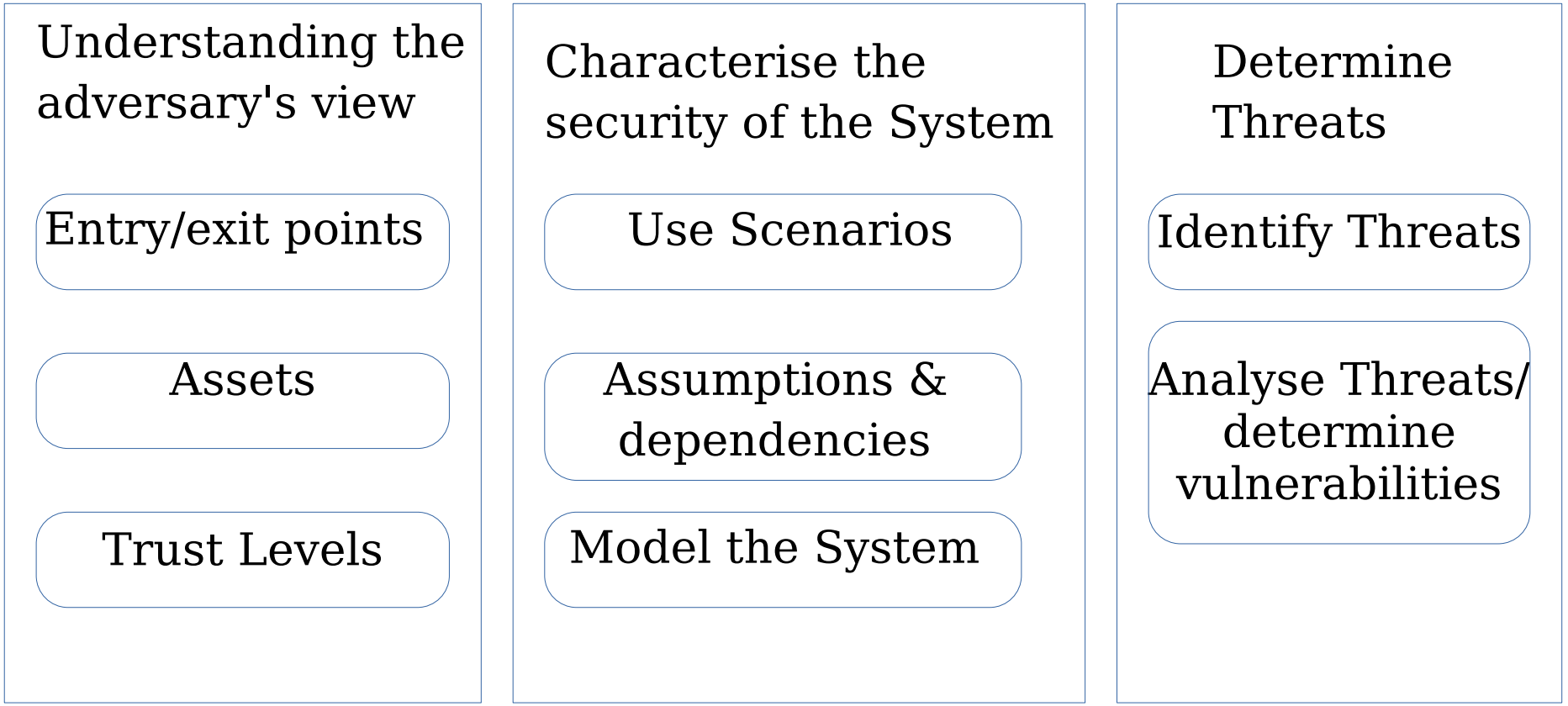
- Damage potential
- Reproducibility
- Exploitability
- Affected Users
- Discoverability

Terminology

- Entry-point & exit point
- Asset
- Threat
- Vulnerability
- Attack / exploit
- Mitigation
- Countermeasure

Process

- Diagram from p30, “Threat Modeling” book, Swiderski&Snyder



Pros & Cons of STRIDE/ DREAD Threat Model

- Pro
  - Detailed analysis – the only time it will probably happen
  - Starts with threats & searching for weaknesses – attackers will follow similar structure
  - Ultimate benefit it that it gets development team seriously thinking about security of their apps – at design time, when it is easy to fix bugs
- Con
  - Time consuming
  - Probably not a glamorous part of the work
    - Some developers find it un-interesting (but some find it fascinating!)
  - Threat model is distinct from code – as code is updated, will threat model evolve
    - Look at traditional design documentation to see possible outcome

The Safety Case For Applications

- Engineers designing mass-transit systems in cities must make a “safety case” to safety experts to convince them that all appropriate measures have been taken, to achieve safety certificate
- Unless shown to be otherwise, assume each application is inherently unsafe
  - Full of security flaws
  - It probably is!
- If you are the security officer for a company purchasing software, you need to be convinved that what you are about to place on your network is safe
- Will become a more frequent task for developers

Security Model Document

- Document clearly outlining the safety case for an application
  - Threat analysis
  - Countermeasures
  - Monitoring (what happens when you app is attacked)
  - Security Reviews
  - Other
- With this document, can go to the security officers of potential customers and help convince them they should approve the safety case for your app

Agile Software Development

- Agile software development is becoming very popular
- Agile principles
  - Non-perscriptive
  - Single representation (code)
  - “Just enough” formality
  - Incremental
  - Delivering business value at frequent intervals
  - Avoid duplication
  - Responding to rapid change

Threat Modelling & Agile Software Development

- Some consider threat models to be “heavy-weight”
- How to develop secure software, in an agile manner?
- Apply agile principles to threat modelling

Interesting Idea

- We spend all this time building application threat models for the development lab
- When our app is deployed on a customer site, wouldn't it be interesting when a real attack occurs, to see that attack in realtime being visualised/explained using the same threat models (words, trees, DFDs) we created
- Hooked up to your Intrusion Detection System
    - “Live” view helps everyone
- Hard work has additional benefits
- Highlights anything that we missed
- No current tools do this

Concluding Thoughts

- Information security is an ongoing process, not a singular event
  - After initial wave of enthusiasm, need to maintain vigilance as software systems evolve
  - Need to continuously improve
  - Don't worry - the attackers will be improving also
- Information security is an enabler for the primary business service, but is (rarely) the primary service itself
  - Very important, but should not dominate thinking
  - You have to still deliver the primary business service