Assistant professor at Eurecom since 2011
Doing security research
- Embedded systems (MCU, smart phones,...)
- Software security (incl. HW support for SW sec)
- Wireless/wired network
- Telecom/telephony security and Fraud
For more details check our group's page: http://s3.eurecom.fr
For (mostly tech) news you can follow me on twitter: @aurelsec
About

• My office is in room 385
  – Down below on the left side :)

• Door Protocol:
  – If you plan to pass by try to drop me an email first
  – Door is (almost) always open, this do not means I’m available (kindly ask before entering)
  – If the door is closed this (often) means I’m busy or away, you can still give a try to knock on the door

• Some projects for the semester are on sifi:
  – If you are interested in doing one on an other topic let me know…
  – I encourage you to find a topic by yourself
  – You can always ask me
Questions?

• Prefer to ask questions in class:
  – I (usually) don’t bite
  – There are no stupid questions (at least if you were not sleeping in class the past hour…)
  – Sometimes accent/language/explanation is not clear: ask for clarification!
  – Don’t be shy: you are probably not the only one with the question
  – Sometimes I may just be wrong (hopefully not too often)
  – If you are shy ask during the break
  – Feel free to interrupt me anytime (but not every minute…)
  – Please help me to make the class interactive
Welcome to the SysSec course

• This is an introductory course that aims to make you “security-aware”

• So far, as a engineers, you have learned to write code and build applications…
  … we now show you how to break them 😊

• Our aim is to help you to understand complexity of current systems
  – learn typical and common security mistakes
  – showing how to break systems
The goal of this course is not (only) to stuff your brains with lots of technical attacks.

But to teach you to **think as an attacker**:
- This is a necessary state of mind in security.
- One can't secure a system without being aware of ways to break it...

**B. Schneier “Law”**

“Any person can invent a security system so clever that he or she can't imagine a way of breaking it.”

See also:

http://www.schneier.com/blog/archives/2008/03/the_security_mi_1.html
OK, but Why?

- In computer science education, you learn to design and program code, but security education falls short
  - Simple programming mistakes lead to serious security problems
  - Today, failing to protect yourself and not being security-aware can be very costly
  - The number of security-related incidents on the Internet increasing fast
  - And by well funded organizations (Stuxnet...)
  - Attribution is difficult, people can easily be falsely accused of performing illegal activities because their computers were hacked
Some Interesting Numbers

• Adware industry is worth several billion dollars per year
• Malware industry is worth 105 billion dollars per year
• 50%-80% of computers connected to Internet are infected with spyware
• 81% of emails is spam (Symantec report Feb 2011)
• 90% of web applications are vulnerable (Cenzic report 2009)

• Cyber Security market in 2011 was worth $63 billion
  – was expected to grow to about $120 billion by 2017
    (Marketsandmarkets.com June 2012)
  – expected to be $231 Billion by 2022
    (Marketsandmarkets.com July 2017)
Some Interesting Numbers

- Governments are now spending a lot in “Cyber” (defense/offense)

- NSA Budget is 10 Billion USD / Year
  - Equivalent to the annual budget of Tunisia

- More than 100,000 employees in USA intelligence agencies
What we expect from you

• Technical interest for security issues
  (Doing security without being interested… is useless)

• Interest in understanding how things work, often from a very low-level point of view
  (If you are scared of binary codes... syssec is not for you)

• Basic programming knowledge and experience
  – Informally courses such as SoftDev or OS are “prerequisites”

• Lot of patience
  (security exercises aren’t like Hollywood scenes 😊)
Administrative Issues

• Mode
  – Lectures covering different practical security aspects
  – Security challenges (e.g., cracking web applications, using security tools, stack-based buffer overflows,...)
  – Ideally one challenge every 2 weeks
    • The challenge system will be deployed soon
    • There will be one Lab session to help you to start/setup
    • Challenges will be part of the final grade, do them!
  – Written final exam (February)

• Slides and News (please visit regularly!)
  – http://s3.eurecom.fr/~aurel/
    (you can find this link through my EURECOM page)
SysSec and Forensics courses

• Courses organization:
  • **SysSec in fall** (A. Francillon) <= you are here!
    • Long course presenting all the basis of system and network security
    • Network security, Memory corruption, web security, OS Security...
  – **Forensics in spring** (D. Balzarotti)
    • Long course
    • Focusing on advanced topics
    • Show students the current (both from a technical and a research) perspective of the fight against cyber-crime
  – Almost no overlapping of topics
  – Different types of homeworks
Topics we will likely cover (but this changes along the road)

1. Host security
   - Unix / Windows / OSX security overview (3h)
   - Race conditions, memory corruption exploitation (2*3h)
   - Trusted computing (3h)

2. Network security (2*3h)
   - Wired / wireless
   - Protection

3. Telephony fraud and abuse (3h)

4. Web security and vulnerabilities (2*3h)

5. Software testing (i.e., finding vulnerabilities) (3h)

6. Malware overview (3h)

7. Guest lectures TBD
• Assignments
  – Starting within a couple of weeks
  – 5 challenges (expected, maybe more)
  – Some points at each challenge solved, extra points for the first ones

• Environment
  – One lab session (Oct 20), TA (Marius) will help registering/setting up ssh/with the challenge
  – In general assignments should be mostly solved individually, at home / any computer with Internet connection and ssh enough
  – Do not lose your SSH key (back it up), If I have to manually reset it (and it’s not my fault) I’ll take some points away from the challenges grade.

• Submission
  – Automatic checking with immediate feedback
  – Everything you do is monitored
  – Cheating will be detected and sanctioned
Grading for the labs

• Challenges graded on 25 points
• The written exam has 75 possible points
• Total of 100 points for the course
• You need to have a total of 50 points to pass the course

• This is subject to change, I'll decide on the final rule!
• Do as many labs as you can, interact, attend lectures
  – Final appreciation can tune the grade
  – Not attending lectures is a very bad idea, slides are not self containing/explanatory
  – Only working with the slides will not be enough!
New this year

• This is an experiment:
  – At the beginning of a lecture or after the break, students can present something (tool, test, exploit, demo…) that is related to a previous course
  – This is not mandatory but will give extra points
  – Need to register at least by Wednesdays ...
Printouts?

- No printouts (save the trees!)
  - Unless when useful (some exercises)

- I'll put the final slides on-line the evening after the lecture
  - I'll try sometimes in advance but no promises
  - e.g., Thursday evening
CTFs

- Eurecom CTF group “Nops”
  - Open to anyone,
  - Not part of the class, held by volunteers (profs, PhD students, self organized)
  - Some training sessions every week, some CTFs participation from time to time

- eSAME:
    - There will be a small “training” by ST engineers on Oct. 19 (TBC)
    - Not really a security exercise
  - Ph0wn: Smart Devices CTF: [http://www.esame-conference.org/program/phown-ctf-contest/](http://www.esame-conference.org/program/phown-ctf-contest/)
    - A security exercise, Nov 19
    - Register in advance
    - Different levels of challenges
    - In teams
But first: Shocking news of the week

- I'll often show some “shocking news” from the field at the beginning of each lecture
  - To wake up those that still asleep!
  - Motivate the course / threats
  - We are covering “hot” topics, new stuff every week!
  - Often recent topic that hit the media
News of the World Scandal

- News of the world was a journal with celebrities content
- They always had fresh scoops
- It turns out that they were listening to people voice mail
- Journal was sued and lost the trial
  - Following a boycott by advertisers the journal closed after 168 years of publication

https://www.theguardian.com/uk-news/2014/jun/24/phone-hacking-scandal-timeline-trial
A logo! News of the World => Week
A logo! News of the World => Week
Russian Hackers Stole NSA Data on U.S. Cyber Defense

The breach, considered the most serious in years, could enable Russia to evade NSA surveillance and more easily infiltrate U.S. networks.
• **What do we learn**
  - Don’t take super secret stuff home to do extra work hours
  - Some antivirus solutions grab complete files from your computer and collect them
  - Governments are suspicious against each other
  - It’s probably the 7th case of leak of information from NSA in a few years…
Intro and History
One big problem

• System and network administrators are not prepared
  – Insufficient resources
  – Lack of training

• Intruders are leveraging the availability of broadband connections
  – Many connected home computers are vulnerable
  – Collections of compromised home computers are “good“ weapons (e.g., for distributed denial of service attacks).
    • High speed networking
    • Powerful CPU
    • Always on
Bugs And Failure

• Hardware and software are developed by humans and therefore are not perfect
• A human error may introduce a **bug** (or fault)
• When a fault gets triggered, it might generate a **failure**...

![Exception message from Windows]

• If the fault is “security-related”, it is usually called a **vulnerability**
• When the vulnerability is triggered (exploited) can lead to the **compromise** of the system (or of part of it)
Vulnerabilities

A little bit of history…

- **1960s** - mainframe computers like the MIT’s Artificial Intelligence Lab became staging ground for hackers. *Hacker was a positive term*

- **1970s** - hackers start tampering with phones (the largest network back then)
  - **1972**, John Draper finds that the whistle that comes with the Cap’n Crunch cereal produces a sound at the 2600 Hz (the same used by AT&T to authorize long-distance calls)
  - It is the start of phone phreaking
  - Steve Jobs and Wozniack first business: blue box
A little bit of history…

• **1973** - Bob Metcalfe wrote RFC 602: “The Stockings Were Hung by the Chimney with Care”
  – ARPA computer network is susceptible to security violations
  – “many people still use passwords which are easy to guess: their first names, their initials, their host name spelled backwards, a string of characters which are easy to type in sequence”

• **1980/81** - Two hacker groups form
  – Legion of Doom (US)
  – Chaos Computer Club (DE)

• **1982** - The term “cyberspace” is coined in the novel *Bourning Chrome*
A little bit of history…

- **1983** - The movie Wargames introduces hackers to the public
- **1986** - German hackers penetrate Lawrence Berkeley Laboratory systems and try to obtain secrets to be sold to the KGB
  - Cliff Stoll (a sysadmin at LBL) found an intruder while investigating a 75 cent accounting discrepancy for CPU time
  - He decided to monitor the intruder in order to find out who he/she was and how he was able to gain privileged access
  - The investigation ends with the arrest of Markus Hess in Germany, who apparently worked for the Eastern Bloc
  - The story is published in a book: “*The Cuckoo’s Egg*”
A little bit of history...

• **1988** - The *Internet worm*, developed by Robert T. Morris, brings down the Internet
  – A mistake in the replication procedure led to unexpected proliferation
  – The Internet had to be “turned off”
  – Damages were estimated in the order of several hundred thousand dollars
  – The CERT (Computer Emergency Response Team) is formed

• **1994** - *Kevin Mitnick* attacks the Supercomputer Center in San Diego using a TCP spoofing attack
  – Arrested in 1995 and sentenced to 46 months in prison
A little bit of history…

• **1990** - Operation Sundevil: secret service arrests hackers in 14 U.S. Cities for credit-card theft and telephone and wire fraud
• **1992** - Release of the movie *Sneakers*
• **1993** – The first *DefCon* conference is held in Las Vegas. It is so popular that it will become an annual event
• **1995** – A russian cracker siphon 10M $ from Citibank and transfer the money to banks around the world
• **1995** – The movie *Hackers* is released
• **1999** – The *melissa* worm causes large problems to the email systems
A little bit of history…

• **2000** – ILOVEYOU, a VBScript worm infects millions of computers within a few hours of its release

• **2002** - Bill Gates announces the 'Trustworthy Computing' initiative, a new direction in Microsoft's software development strategy aimed at increasing security

• **2003** – The SQL Slammer worm infected 75,000 machines (90% of the possible targets) in 10 minutes
  – Starts the fear for the *flash worms*

• **2005-2010** – Worms are slowly replaced by botnets

• **2010** – *Stuxnet* attacks centrifuge systems in nuclear facilities in Iran
  – Completely new (and unexpected) level of sophistication
  – State sponsored, cyberwar?
A little bit of history…

• **2013** – Snowden revelations, threat model changes. We are facing an extremely powerful adversary!
  – Big Brother
  – Highly funded
  – Zero days, pervasive network surveillance and injection
• **2016** IoT Botnet make huge DDoS with hundred thousands compromised devices (IP cameras, home routers...)
Changing Nature of the Threat

• Nowadays Intruders are more prepared and organized
• Internet attacks are easy, low-risk and difficult to trace
• Intruder tools are increasingly sophisticated and easy to use (e.g., by Script kiddies)
• The complexity of Internet-related applications and protocols are increasing – and so is our dependency on them
• Malware/attacks is an “industry”
Online Crime is a Business

Klikparty, 2007
Online Crime is a Business

KoobFace Gang

http://nakedsecurity.sophos.com/koobface/
Sometimes a “Legal” business too

- Selling exploits or surveillance tools to (shady) governments
Good and Bad Hackers

• The term “hacker” was introduced at MIT in the 60s to describe “computer wizards”

[...] someone who lives and breathes computers, who knows all about computers, who can get a computer to do anything. Equally important, though, is the hacker's attitude. Computer programming must be a hobby, something done for fun, not out of a sense of duty or for the money.
- Brian Harvey, University of Berkeley

• The term was later associated to “malicious hackers” or “crackers”, that is, people that perform intrusions and misuse computer systems
hacker
A person who delights in having an intimate understanding of the internal workings of a system, computers and computer networks in particular. The term is often misused in a pejorative context, where "cracker" would be the correct term. See also: cracker.

cracker
A cracker is an individual who attempts to access computer systems without authorization. These individuals are often malicious, as opposed to hackers, and have many means at their disposal for breaking into a system. See also: hacker, Computer Emergency Response Team, Trojan Horse, virus, worm.
Terminology

• **Black Hat**: a cracker, someone bent on breaking into the system you are protecting

• **White hat**: usually associated to friendly security specialists

• **Script Kiddie**: lowest form of cracker; script kiddies do mischief with scripts and programs written by others, often without understanding the exploit they are using
Terminology

• What is an attack?
  – no easy answer, it depends

• First: what is the security policy
  – The framework within which an organization establishes needed levels of information security to achieve the desired integrity, confidentiality, and availability goals. A policy is a statement of information values, protection responsibilities, and organization commitment for a system.
    (US Congressional Office of Technology)

  – A set of guidelines defining what you want to protect and what you want to allow at your site.
Terminology

- What you want to protect?
  - defines assets

- What are the goals of your protection efforts?
  - Integrity
    - Data has not been altered or destroyed in an unauthorized manner
  - Confidentiality
    - Information is not made available or disclosed to unauthorized individuals, entities or processes
  - Availability
    - Data/Service being accessible and usable upon demand by an authorized entity
Terminology

• What do you want to protect against?
  – threat model
  – risk analysis

• Different security policies
  – bank answers questions different than home user

• Attack
  – any maliciously intended act against a system or a population of systems
  – any action that violates a given security policy
Threats vs Vulnerabilities

- A **Threat** defines who might attack against what assets, using what resources, with what goal in mind, when/where/why, and with what probability.
- **Vulnerabilities** are specific weakness in security that could be exploited by adversaries with a wide range of motivations and interest in a lot of different assets.

**Threat**: Thieves could break into our facility and steal our equipment.

**Vulnerability**: The lock we are using on the building doors is easy to pick or bump.

**Threat**: Adversaries might install malware in the computer so they can steal social security numbers for purposes of identity theft.

**Vulnerability**: The computer do not have up to date virus signatures.
Malicious hacking/cracking is illegal

However, discussing vulnerabilities and how they are actually exploited is useful to educate and increase awareness.

A full disclosure policy has been advocated by many respected researchers, provided that...

- The information disclosed has been already distributed to the parties that may provide a solution to the problem (e.g., vendors)
  - See: Responsible vulnerability disclosure process (IETF Internet Draft)
- The ultimate goal is to prevent similar mistakes from being repeated
Security Overview
Security Overview

- Security issues at various stages of application life-cycle
  - mistakes, vulnerabilities, and exploits
  - avoidance, detection, and defense

- Architecture
  - security considerations when designing the application

- Implementation
  - security considerations when writing the application

- Operation
  - security considerations when the application is in production
Security Overview

Architecture and design

- validation of requirements (building the right model)
- verification of design (building the model right)

Common problems

- authentication and privileges
  - session replay
  - principle of least privilege
- communication protocol design
  - sniffing, man-in-the-middle
  - session killing, hijacking
- parallelism and resource access
  - race conditions
- denial of service
Security Overview

**Implementation**
- verification of implementation
- classic vulnerabilities (often programming-language-specific)

**Common problems**
- buffer overflows
  - Static: stack-based buffer overflows
  - Dynamic: heap-based buffer overflows
- input validation
  - URL encoding
  - document root escape
  - SQL injection
- back doors
Security Overview

Operation

– decisions made after software is deployed
– often not under developer’s control

Common problems

– denial of service (DOS)
  • network DOS
  • distributed DOS, zombies
– administration problems
  • weak passwords
  • password cracking
  • unsafe defaults
Insecure Software

….or, why good people write bad code

• **Technical factors**
  – complexity of task

• **Economic factors**
  – deadlines
  – insufficient funding

• **Human factors**
  – mental models
  – social factors
Technical Factors

• Complexity
  – algorithmic complexity
  – parallel processes, threads
  – multi-user
  – Indeterminism

• Composition
  – incorrect assumptions
  – surprising interactions

• Changes
  – consequences are hard to predict
  – example: Sun tarballs
    • Small change leads to leaking password hashes
  – Debian RNG: remove “uninitialized” read
Economic Factors

• Production pressure
  – not enough time
  – not enough manpower for testing

• Security is not a feature
  – just secure enough

• Open-source vs. closed-source debate
  – open-source is peer-reviewed
  – closed-source is written by professionals

• Legacy software
Human Factors

• Poor risk assessment
  – invisible enemy

• Mental models
  – only check for errors that are understood
  – assume software is used for a specific task
Improvement

• **Tools**
  – detect mistakes and vulnerabilities
  – support programmer
  – formal verification

• **Standards and metrics**
  – hold vendors accountable
  – allow for comparison between products

• **Education**
  – that’s what we are trying to do here ;-}
Methods of attacking

• Eavesdropping
  – getting copies of information without authorization

• Masquerading (impersonating)
  – sending messages with other’s identity

• Message tampering
  – change content of message
Methods of attacking

• Replaying
  – store a message and send it again later, e.g., resend a payment message

• Exploiting
  – using bugs in software to get access to a host

• Combinations
  – Man in the middle attack
    • emulate communication of both attacked partners (e.g., cause havoc and confusion)
Social Engineering

• “The art and science of getting someone to comply to your wishes”
  – Security is all about trust.
    Unfortunately, the weakest link, the user, is often the target

• Performed in many different forms
  – Social engineering by phone
  – Dumpster Diving
  – Reverse social engineering
  – Malware disguised as fake anti-virus

• According to report, secret services often use social engineering techniques for intrusion
Design and Architectural Principles
Overview

• Security issues at various stages of application life-cycle
  – mistakes, vulnerabilities, and exploits
  – avoidance, detection, and defense

• Architecture
  – security considerations when designing the application

• Implementation
  – security considerations when writing the application

• Operation
  – security considerations when the application is in use
Microsoft SDL

- **Training**
  - Core training

- **Requirements**
  - Define quality gates/bug bar
  - Analyze security and privacy risk

- **Design**
  - Attack surface analysis
  - Threat Modeling

- **Implementation**
  - Specify tools
  - Enforce banned functions
  - Static analysis

- **Verification**
  - Dynamic/Fuzz testing
  - Verify threat models/attack surface

- **Release**
  - Response plan
  - Final security review
  - Release archive

- **Response**
  - Response execution

**Tools**

- SDL Process Template
- MSF-Agile+SDL Process Template
- SDL Threat Modeling Tool
- SDL Process Template
- MSF-Agile+SDL Process Template
- Banned.h
- SiteLock ATL Template
- FxCop
- Code Analysis for C/C++
- Anti-XSS Library
- CAT.NET 32-bit
- CAT.NET 64-bit
- SDL Process Template
- MSF-Agile+SDL Process Template
- BinScope
- MiniFuzz
- SDL Regex Fuzzer
- AppVerifier
- SDL Process Template
- MSF-Agile+SDL Process Template

**Experience this Site in Silverlight. Download the FREE Silverlight plug-in →**
Architecture – A Quick Recap

- **Software architecture**
  A *representation of an engineered software system, and the process and discipline for effectively implementing the design(s) for such a system*

- **Representation**
  - architecture concerned with components and their relationships

- **Process**
  - steps are provided that describe how to change design within set of constraints

- **Discipline**
  - set of principles how to design system within constraints
Software architecture has emerged as a crucial part of the design process. Much work was done in the early 90s, and today, there are research issues such as product family architectures, architectural description languages, flexibility, fault tolerance, etc.

Software architecture encompasses the structures of large software systems. The architectural view is abstracted, mostly concerned with interface descriptions (behavior), and distills details of implementation (such as algorithmic aspects and data representation).
• What is a security architecture?

A *body* of high-level *design principles* and decisions that allow a programmer to say "Yes" with confidence and "No" with certainty.

A *framework* for *secure design*, which embodies the four classic stages of information security: *protect*, *deter*, *detect*, and *react*.

• Security is a measure of the architecture’s ability to resist unauthorized usage
  
  – at the same time, services need to be provided to legitimate users
What happens if architecture is flawed?

• Some history: The Swedish warship Vasa
  – now in Stockholm, Vasa Museum
  – a solemn reminder for engineers
  – the ship was built well, but its architecture was \textit{flawed}.
  – on its first voyage, a bit of wind, and …

• So what does Vasa have to do with security?
  – your code might be engineered well, but if your architecture is bad from a security point of view, your system may be broken by attacker
Vasa Today
Cost of fixing security flaws during different development phases

- Design: (cost = 1)
- Implementation: (cost = 6.5)
- Testing: (cost = 15)
- Post-Release: (cost = 60)
Security and Design

• Systems are often designed without security in mind
  – application programmer is often more worried about solving the problem than protecting the system
  – often, security is ignored because either the policy is generally not available, or it is easier to ignore security issues

• Organizations and individuals want their technology to survive attacks, failures and accidents
  – critical systems need to be survivable
Design Principles

• Design is a complex, creative process
• No standard technique to make design secure
  – But general rules derived from experience

• 8 principles according to Saltzer and Schroeder (1975)
  “The protection of information of computer systems”
  – Economy of Mechanism
  – Fail-safe defaults
  – Complete mediation
  – Open design
  – Separation of privilege
  – Least privilege
  – Least common mechanism
  – Psychological acceptability
Economy of Mechanism

• Design should be as simple as possible
  – KISS -- keep it simple, stupid
  – Brian W. Kernighan
    “Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.”

• When things are complex, users get them wrong
Fail-safe Defaults

• **Allow** as default action
  – grant access when not explicitly forbidden
  – in case of mistake, access allowed (often not noticed)
  – improves ease-of-use
  – wrong psychological model

• **Deny** as default action
  – grant access only on explicit permission
  – in case of mistake, access denied (noticed quickly)
  – improves security
  – important for firewall configurations and input validation tasks
Fail-safe Defaults

- **Configuration**
  - secure initial configuration
  - easy (re)configuration

- **Secure initial configuration**
  - no default passwords
  - no test users
  - files are write-protected, owned by root/admin

- **Error messages**
  - should be very generic
  - additional information in log files
Complete Mediation

• Complete access control
  – check every access to every object
  – include all aspects (normal operation, initialization, maintenance, ...)
  – caching of checks is dangerous
  – identification of source of action (authentication) is crucial

• Trusted path
  – make sure that user is talking to authentication program
  – important for safe login (thwart fake logins)
  – Windows “control-alt-delete” sequence
Complete Mediation

• Secure interface
  – minimal
  – narrow
  – non-bypassable (e.g., check at server, not client)

• Input validation

• Trust input only from trustworthy channels
  – any value that can be influenced by user cannot be trusted
    • do not authenticate based on IP source addresses / ports
    • E-mail sender can be forged
    • hidden fields or client side checks are inappropriate
  – safely load initialization (configuration)
Open Design

• Design must not be secret
  – security mechanisms must be known
  – allows review
  – establishes trust
  – unrealistic to keep mechanism secret in widely distributed systems

• Security depends on secrecy of few, small tokens
  – keys
  – passwords
Open Design

• **Kerckhoff's principle** for cryptography:
  “A cryptosystem should be secure even if everything about the system, except the key, is public knowledge”

• Don't rely on secrecy does not mean make everything public

• Companies often keep secret the details of a system
  – Security through Obscurity
  – May improve security in the short term, but it is generally a bad idea on the long run
Separation of Privilege

• Access depends on more than one condition
  – for example, two keys are required to access a resource
  – two privileges can be (physically) distributed
  – more robust and flexible

• Classic examples
  – launch of nuclear weapons requires two people
  – bank safe

• Related principle
  – compartmentalization
Separation of Privilege

• Compartmentalization
  – break system in different, isolated parts and minimize privileges in each part
  – don’t implement all-or-nothing model
  → minimizes possible damage

• Sandbox
  – traditional compartmentalization technique
  – examples
    • Java sandbox (bytecode verifier, class loader, security manager)
    • virtual machines
    • Rendering in google Chrome
    • System jails (chroot)
Least Privilege

• Operate with least number of rights to complete task
  – minimize damage
  – minimize interactions between privileged programs
    • reduce unintentional, unwanted use

• Minimize granted privileges
  – avoid *setuid* root programs (UNIX/Linux)
    • use groups and *setgid* (e.g., group *games* for high scores)
    • use special user (e.g., *nobody* for web server)
  – make file owner different from *setuid* user
    • taking control of process does not allow to modify program images
Least Privilege

• Minimize granted privileges
  – database restrictions
    • limit access to needed tables
    • use stored procedures

• Minimize time that privilege can be used
  – drop privileges as soon as possible
  – make sure to clear saved ID values

• Minimize time that privilege is active
  – temporarily drop privileges
  – can often be re-enabled by the attacker, but still
    protects against some kinds of attacks (e.g., file access)
Least Privilege

• Minimize modules that are granted privilege
  – optimally, only single module uses privileges and drops them
  – two separate programs
    • one can be large and untrusted
    • other is small and can perform critical operations
    • important for GUI applications that require privileges

• Limit view of system
  – limit file system view by setting new root directory
    chroot() – on Unix
  – more complete virtual machine abstraction
    BSD system call jail(2)
  – Honeypot
Least Privilege

• Do not use \textit{setuid} scripts
  – “race condition” problems
  – Linux drops \textit{setuid} settings

• Minimize accessible data
  – CGI scripts
    • place data used by script outside document root

• Minimize available resources
  – quotas

• Paper: Provos et al., \textit{Preventing Privilege Escalation},
  \textit{12th USENIX Security Symposium}, 2003
Least Common Mechanisms

• Minimize shared mechanisms
  – reduce potentially dangerous information flow
  – reduce possible interactions

• Problems
  – beware of “race conditions”
  – avoid temporary files in global directories
Psychological Acceptability

• Easy-to-use human interface
  – easy to apply security mechanisms routinely
  – easy to apply security mechanisms correctly
  – interface has to support mental model
    • do what is expected intuitively (e.g., personal firewalls)

• Authentication
  – passwords
    • enforce minimum length (what is the minimum length?)
    • enforce frequent changes
  – PKI (public key infrastructure)
    • overhead vs. security
One more Design Principle

- Separate data and control
  - failed separation is reason for many security vulnerabilities
    - from buffer overflows to macro viruses
  - distinction between control information and data has to be clear

- Problematic
  - with automatically executing code in data files
    - JavaScript in web pages ("eval")
    - automatic preview of web pages in emails
    - macros in Word
  - when using mobile code
    - code that is downloaded and executed locally
Practice Defense in Depth

• Have several layers of security
  – Preventing is not enough, you also need detection and mitigation mechanisms
  – Two controls are better than one

• No single point of failure
"The only system which is truly secure is one which is switched off and unplugged, locked in a titanium-lined safe, buried in a concrete bunker, and surrounded by nerve gas and very highly paid armed guards. Even then, I wouldn't stake my life on it"

-- Gene Spafford
Minimize Attack Surface

• Minimize
  – number of open sockets
  – number of services
  – number of services running by default
  – number of services running with high privileges
  – number of dynamic content webpages
  – number of accounts with administrator rights
  – number of files & directories with weak access control

• Minimize the “time” surface
  – Automatically lock screen after n minutes
  – it’s good practice to zero-out memory that contains sensitive information (usually, decrypted information) as soon as it’s no longer needed (sun tarball example)
Retrofitting Applications

• Applying security techniques to existing applications
  – element of overall system design
  – when no source code available or
  – complete redesign too complicated

• Wrappers
  – move original application to new location and replace it with small program or script that
    • checks (and perhaps sanitizes) command-line parameters,
    • prepares a restricted runtime, and
    • invokes the target application from its new location
    – can provide logging
    – can provide possibility for prologue and epilogue code
Retrofitting Applications

• Example wrappers
  – AusCERT Overflow Wrapper
    • exits when any command line argument exceeds a certain length
  – TCP Wrappers
    • replaces inetd (for telnet, ftp, finger, …)
    • access control
    • logging
  – sendmail restricted shell (smrsh, replacement for /bin/sh)
    • sendmail known for security problems
    • smrsh restricts accessible binaries
    • interestingly, was vulnerable to two exploits that allow arbitrary code execution
Retrofitting Applications

• **Interposition**
  – insert program that we control between two pieces of software that we do not control
  – filtering of data
    • add security checks and constraints
  – network proxy
    • application policy enforcement
    • SYN flood protection
  – input sanitization
Bad Practice

• Being too specific too soon
  – without having a design, solve technical problems and start implementation

• Focus only on functionality
  – security must be built in from the beginning

• Not considering economic factors
  – ignoring the cost of security features
Bad Practice

- Not considering the human factor
  - propose solutions that users strongly dislike
    - biometric scanners instead of passwords
  - propose solutions that are annoying
    - change passwords too frequently
    - terminate idle sessions too fast
  - propose solutions that require considerable additional effort
    - producing too many alerts (e.g., snort -- “useless”)
    - require checking of many different log-files
Conclusion

• We looked at introductory topics
  – Social engineering, passwords, importance of security

• We discussed architectural considerations and issues

• Next week Lecture on host security
  – Windows/Linux security