Malware & Software Security

Today’s Plan

- 09:00 – 10:00: **MALWARE SECURITY (William Stallings, Lawrie Brown, *Computer Security Principles and Practice* [Chapter 6], Third edition, Pearson, Australia) & SOFTWARE SECURITY (Various Sources).**

- 10:00 – 10:30: Demo: Lab 7 Completion.

- 10:30 – 12:00: Coursework.

Malware Security

What are we fighting against?

- Propagation Mechanism:
  - Infection of existing files (viruses).
  - Exploitation of network vulnerabilities (worms).
  - Social engineering attacks (trojans).
  - Blended attacks (multiple methods of propagation).
  - Advanced Persistent Threats (careful target selection).
- Types of Payload:
  - Data destruction.
  - Data kidnapping.
  - Real-world damage.
  - Logic bombs.
  - Bots/Zombies.
  - Information theft (phishing).
  - Stealthing (rootkit/backdoor).
  - Cryptojacking.

Prevention

- Hardening the system.
- Four main elements of prevention:
  - Policy
  - Awareness
  - Vulnerability mitigation
  - Threat mitigation
- Update systems.
- Set appropriate access controls (reduce accessible files by any user).
- User awareness training (against social engineering attacks).

Effective Malware Countermeasures

- **Generality**: Wide variety of attacks.
- **Timeliness**: Quick response.
- **Resiliency**: Resistant to evasion techniques.
- **Minimal denial-of-service costs**: Should not disrupt normal operations.
- **Transparency**: Should not alter legacy OS, SW and HW.
- **Global and local coverage**: Deal inside and outside network.

- The more are used, more in line with the **defence-in-depth** principle.

Technical Mechanisms

- Once attack occurs, support by:
  - **Detection**: Determine that attack has occurred and locate malware.
  - **Identification**: Specific malware.
  - **Removal**: Eliminate all traces.

- Detection but no Identification/Removal: Backup version.

- Three places to detect:
  - In host (host-based scanner).
  - In network (firewall).
  - In general (distributed mechanism).

Host-Based Scanner

- Four generations:
  - **Simple scanner**: Requires a malware signature or a length change identification.
  - **Heuristic scanner**: Searches for probable malware instances based on rules.
    - Encryption detection.
    - Integrity checking.
  - **Activity traps**: Memory-resident programs that identify malware by its actions.
  - **Full-feature protection**: Various techniques used in conjunction.

4-Gen: Full-feature Protection

- **Generic Decryption (GD)**: Fast scanning of polymorphic virus.
  - **CPU Emulator**: Instructions are run first in a small VM to test.
  - **Virus Signature Scanner.**
  - **Emulation Control Module**: Interrupts interpretation to scan the target code.
- **Which is the most difficult thing to implement in GD?**
- **Host-Based Behaviour-Blocking System**: Integrates within OS to monitor:
  - Attempts to -rwx files.
  - Attempts to format disk.
  - Modification to logic of files and macros.
  - Modification of critical systems (i.e. start-up).
  - Scripting of e-mail and instant messaging.
  - Initiation of network communications.
- **Which is the main advantage of this approach?**
- **Which is the main limitation of behaviour-blocking alone?**

4-Gen: Full-feature Protection

- **Spyware Detection and Removal:**
  - Specific detection and removal.
- **Rootkit Countermeasurements:**
  - Requires network and computer security.
  - Look for behaviours that may indicate rootkit presence (interception of systems calls, keylogger activity, etc).
  - File integrity check.
  - If kernel-rootkit is detected, entire OS is reinstalled.

Perimeter Scanning Approaches

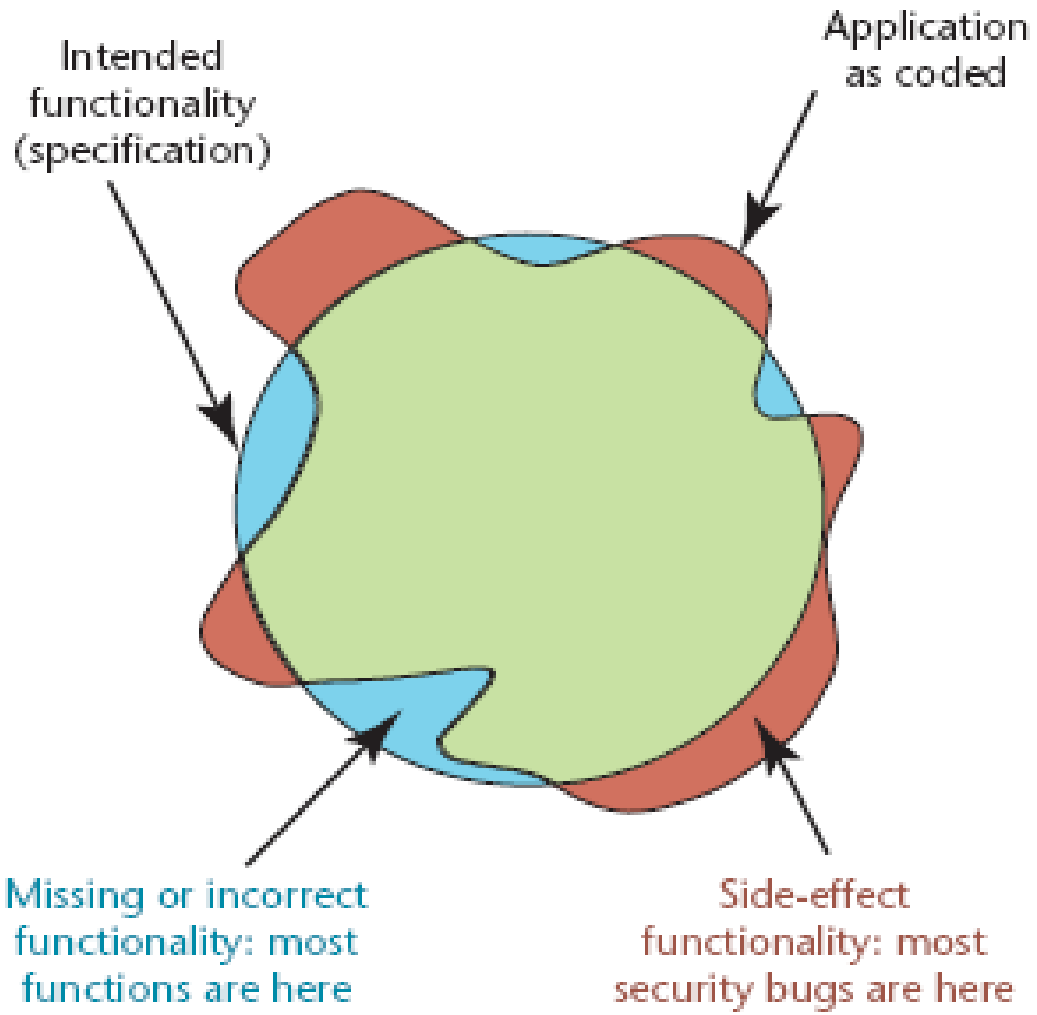
- **Ingress monitors:**
  - Border between the enterprise network and the Internet.
  - Part of:
    - The ingress filtering software.
    - Border router.
    - External firewall or a separate passive monitor.
  - Can use either anomaly or signature and heuristic approaches to detect malware traffic.
- **Egress monitors:**
  - Exit of individual LANs on the enterprise network or border between the enterprise network and the Internet.
  - In the former, the monitor can be part of the filtering software of a LAN router or switch.
  - The two types can be installed in one device.
  - Designed to catch the source of a malware attack by monitoring outgoing traffic.
  - Look for the common sequential or random scanning behaviour used by worms and rate limit or block it.
  - Detect and respond to abnormally high email traffic.

Distributer Intelligence Gathering Approaches

- The final location where anti-virus software is used is in a distributed configuration.
- It gathers data from a large number of both host-based and perimeter sensors, relays this intelligence to a central analysis system able to correlate and analyse the data, which can then return updated signatures and behaviour patterns to enable all of the coordinated systems to respond and defend against malware attacks.
- A number of such systems have been proposed.
  - Distributed intrusion prevention system (D-IPS) targeting malware.

Software Security

Intended vs Implemented SW Security



H. H. Thompson, Security Innovation

Types of Software Flaws

Non-intentional. These can be:

- **Validation flaws**. The code fails to check for valid input data.
- **Domain flaws**. This is where data leaks from one program to another.
- **Serialisation flaws**. This is where data changes while being passed from one program to another.
- **Identification/Authentication flaws**. This is where there is a lack of identification for processes or users.
- **Boundary condition flaws**. This is where resource access is not checked, and can thus allow an external hacker to use up resources.
- **Logic flaws**.

Bug

Trap-door

Intentional. This can either be caused by malious code (such as a Trojan or back-door programs).

Author: Prof Bill Buchanan

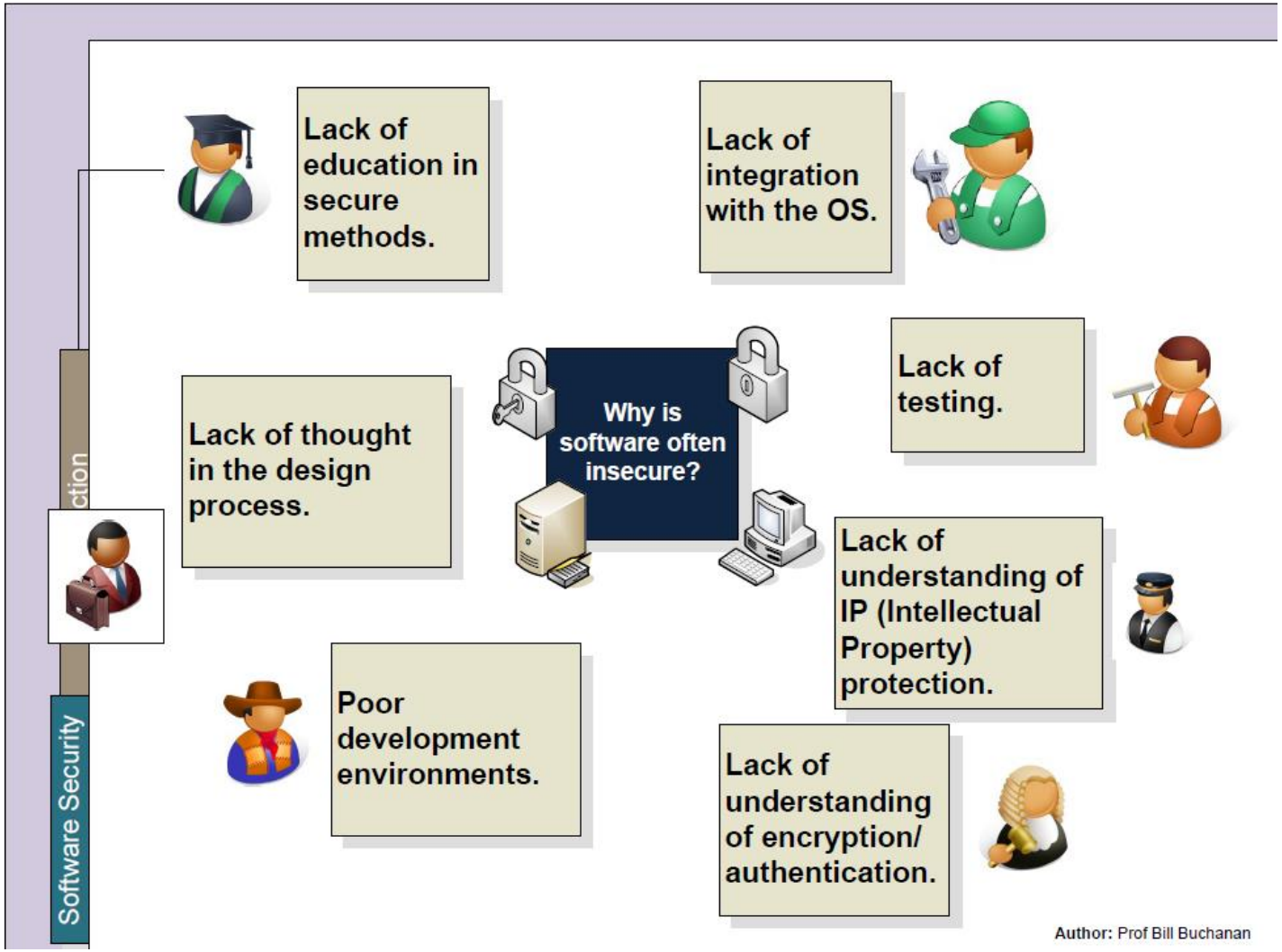
Bad Software

- Software that exposes confidential data to unauthenticated users.
- Software which crashes or grinds to a halt when exposed to faulty inputs.
- Software which allows an attacker to inject code and execute it.
- Software which executes privileged commands for an attacker.

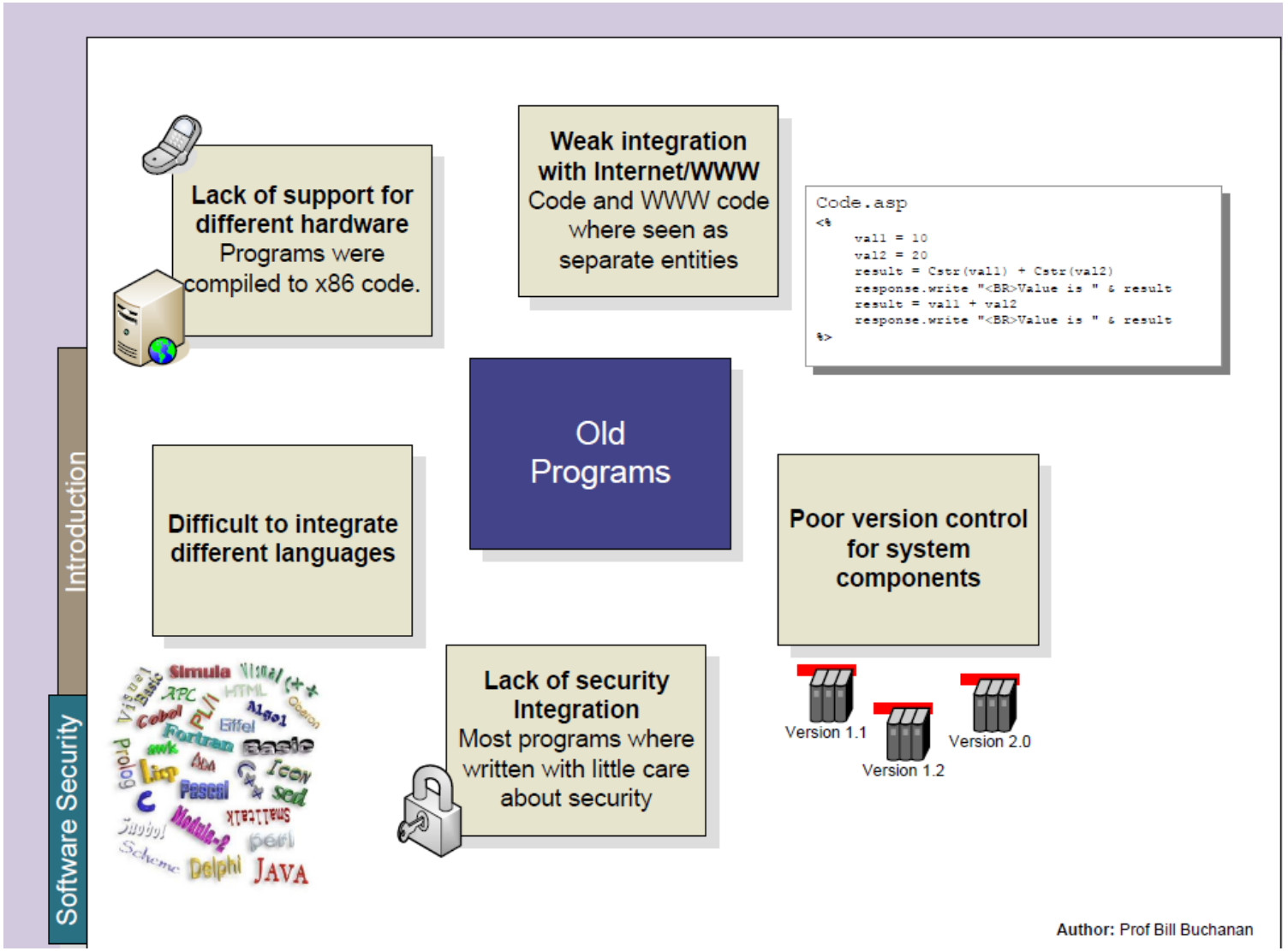
Software Insecurity

- Networked software is not designed to withstand a hostile environment.
- Development tools do not prevent simple security bugs (i.e., buffer overflows).
- QA Testing methods do not address security.
- Customers pay for bad software.

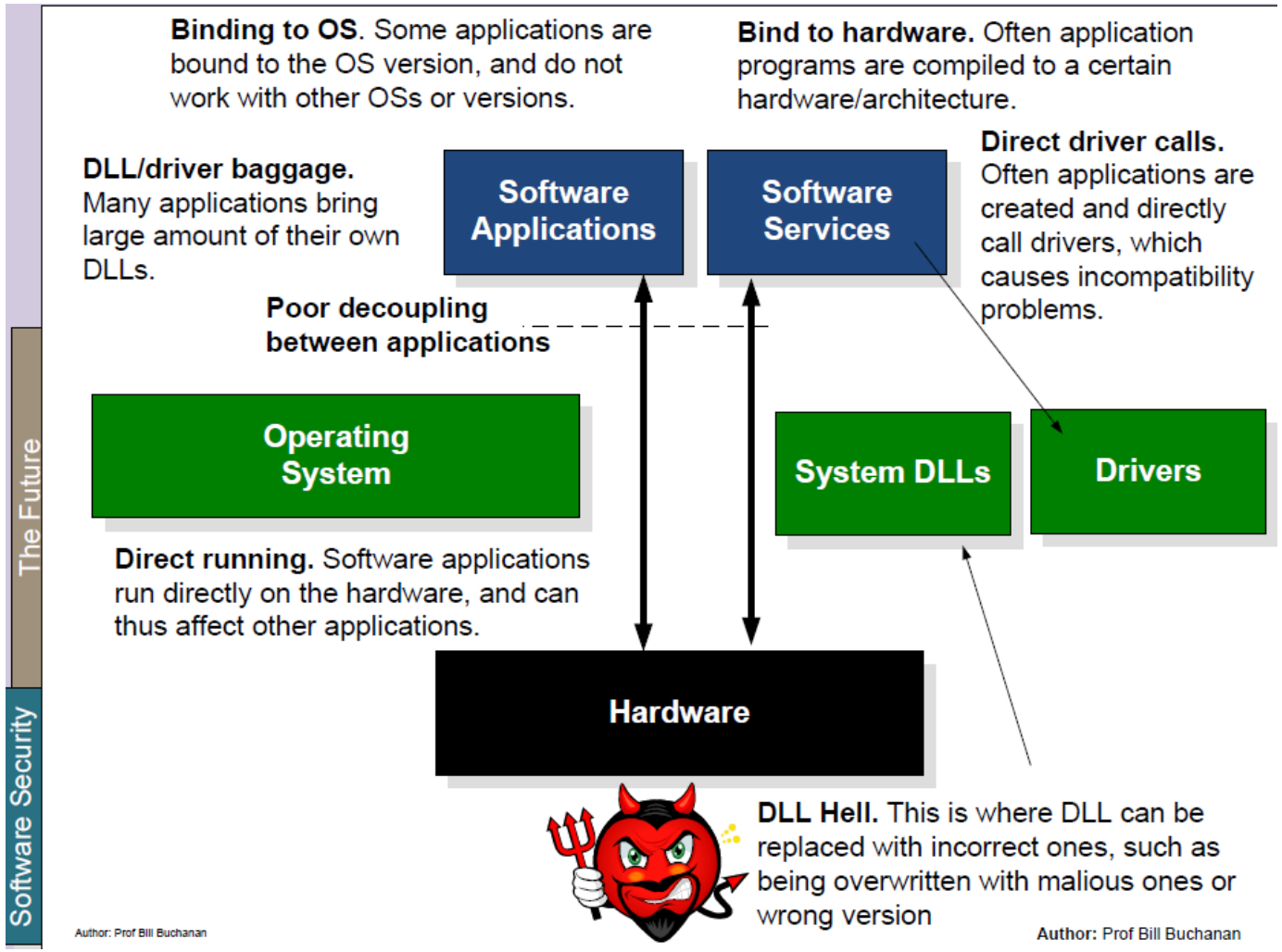
Software Insecurity



Software Insecurity



Software Security Assurance



Software Security Questions

- What are the methods and technologies, available to us if we want to provide security?
  - Security in the software development lifecycle.
  - Engineering & design principles.
  - Security technologies.
- What are the methods and technologies available to the enemy who wants to break security ?
  - What are the threats and vulnerabilities we're up against?
  - What are the resources and tools available to attackers?

Software Security Examples

- Cryptography
  - For threats related to insecure communication and storage.
- Access control
  - For threats related to misbehaving users.
- Language-based security
  - For threats related to misbehaving programs:
    - Typing
    - Memory-safety
    - Sandboxing

Software Error Categories

1. **Insecure interaction between components:**
   - Improper neutralisation of:
     - Special elements used in commands (OS commands, SQL commands, etc.).
     - Input during web page generation.
   - Unrestricted upload of files.
   - URL redirection to untrusted sites.

Software Error Categories

2. Risky resource management:

- Buffer copy without size checking.
- Improper limitation of a pathname.
- Download code without integrity check.
- Inclusion of functionality from untrusted resources.
- Use of potentially dangerous functions.
- Incorrect calculation of buffer size.
- Uncontrolled format string.
- Integer overflow.

Software Error Categories

3. Porous defences:

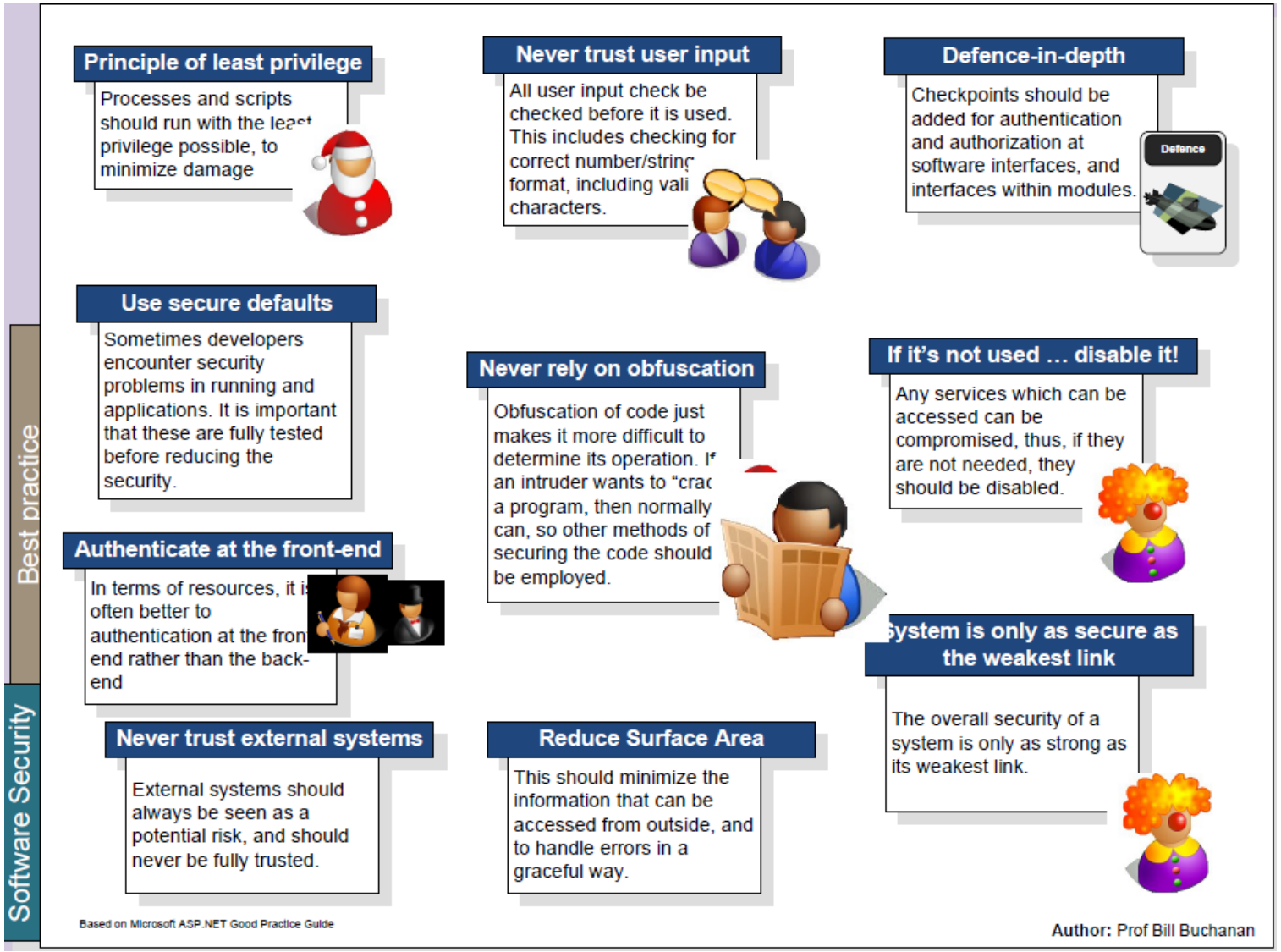
- Missing authentication for critical function.
- Missing authorisation.
- Use of hard coded credentials.
- Missing encryption of critical data.
- Reliance on untrusted inputs in security decisions.
- Execution with unnecessary privileges.
- Use of broken or risky crypto algorithms.
- Improper restrictions on authentication attempts.

Software Error Categories

4. Handling Input:

- Input Size and Buffer Overflow.
- Interpretation of Program Input.
  - Binary vs. text
  - Injection Flaws:
    - Command injection (e.g. `finger*`)
    - SQL injection (`SELECT * FROM … where …`)
    - Code, mail, format string, interpreter injection
- Validating Input Syntax:
  - Regular expressions
  - ASCII Unicode: encoding of "`/`" is "`2F`" and ""`C0AF`"
- Input Fuzzing (randomly generated data).

Software Security Best Practices



Demo

Lab 7 Completion