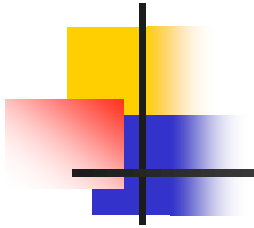


CSE 465
Information Assurance

IA in Outsourcing and Open-Source Software

Professor Stephen S. Yau

What is Outsourcing?

- An act ***of delegating or transferring some or all of the business process and services to external providers***
- External providers will develop, manage and/or administer these services in accordance with the contractual agreements on deliverables and QoS
- Examples of outsourcing
  - Using cloud computing services
  - Outsourced software development
  - Using existing software from other organizations
  - Using Commercial Off-The-Shelf (COTS) items

Benefits of Outsourcing

- Low development cost
- Quicker development and deployment of timely services or products (accelerated time-to-market)
- Low cost for operations, maintenance and updates
- Accessing state-of-the-art expertise and skilled services
- Concentrating and investing more on core business

IA Challenges in Outsourcing

- ***Risk analysis and risk mitigation strategies*** are ***very difficult*** for outsourcing development
- ***Lose control over the process*** of outsourced software development
- Client and outsourcing provider may have ***conflicting security policies and procedures***.
- Creating ***risks to clie***nt's ***intellectual property***

IA Challenges in Outsourcing (cont.)

- Outsourcing providers' facilities and all personnel ***may not adhere to the client's standards and laws*** regarding protection of data and intellectual property?
- Clients and outsourcing providers may not both include ***security professionals*** in regular review meetings to ensure satisfaction of security requirements, and enforcement of security policies and procedures without conflict"
- Clients and outsourcing providers ***may not document all the development process activities*** in proper format and reviewed carefully?

COTS-Based Systems

- COTS: ***Commercial Off-The-Shelf***
- Companies, organizations and government agencies often use COTS items to build a system
  - Example: Use commercial database management software and web server software to build a web based system
- Benefits of COTS-based systems
  - Reduce development cost and time
  - COTS are proven to work
  - Technical supports from vendors

Risks for COTS-Based Systems

- ***Difficult to verify security*** of COTS products
- COTS software is generally a more ***attractive target for attackers*** than customized code
  - COTS software may be well known and widely available
  - More information on security vulnerabilities and viable attack patterns is shared
  - Attackers likely gain more benefits from attacking COTS products
- COTS vendors have very ***limited liability***
- COTS components are often ***generic***
- Involved DoD agencies: Defense Information Systems Agency (DISA), Joint Interoperability Test Command (JITC)

Mitigating Risks for COTS-Based Systems

- ***Identify all components*** to be integrated into COTS-based systems, including both COTS and customized components.
- ***Understand business goals and context*** of the system
  - What ***sensitive information*** is processed and stored in system?
  - What ***security mechanisms*** are required to protect the sensitive information in the system?
- ***Understand how COTS and customized components are connected***
- ***Control Access***
  - Developers of COTS components generally assume that access is controlled in appropriate ways by distributers and/or users

Mitigating Risks for COTS-Based Systems (cont.)

- ***Ask the Vendor***
  - Security-related problems over the history of COTS components
  - Security-related patches for the problems
- ***History***: Frequency of occurrences of security-related problems and vendor's diligence in addressing the security problems are important factors in selecting the COTS components.
- Engage with ***user community and security community***
- Engage with the ***experts***
- Look for ***certification***

Open-Source Software (OSS)

- OSS is software whose source code and certain rights normally reserved for copyright holders are ***freely available to public for redistribution, modification and examination***
- OSS has become more and more popular due to
  - Internet
  - Various software development tools which help OSS development
  - Prevent predatory vendors lock-in (avoid the situation in which customers depend on a single vendor for some product).

Examples of OSS

- Apache HTTP Server (http://httpd.apache.org/) (web server)
- GNOME (http://www.gnome.org/) (Linux desktop environment)
- GNU Compiler Collection (http://www.gnu.org/software/gcc) (GCC, a suite of compilation tools for C, C++, etc)
- KDE (http://www.kde.org/) (Linux desktop environment)
- Mozilla (http://www.mozilla.org/) (web browser and email client)
- Firefox (http://www.mozilla.com/en-US/firefox/) (web browser based on Mozilla)
- MySQL (http://www.mysql.com/) (database)
- OpenOffice.org (http://www.openoffice.org/) (office suite, including word processor, spreadsheet, and presentation software)
- PHP (http://www.php.net/) (web development)
- Ruby(http://www.ruby-lang.org/en/) (programming language)

Characteristics of OSS Development

- ***Collaborative development*** in public open communities
- ***Sharing ideas, technologies and expertise***
- Distributed and independent ***peer reviews***
- ***Better quality and higher reliability*** Lower development ***cost***
- Users treated as ***co-developers***, reporting bugs and bug fixes.
- ***Early releases***
- ***Frequent integration***
- ***Multiple versions*** with different features
- ***Beta versions*** with latest features and risks of having more vulnerabilities
- ***Stable versions*** with fewer features that have been thoroughly tested.
- ***High modularity and flexible structure***

Open Source Initiative (OSI)

- OSI (https://www.opensource.org/) is a non-profit organization founded in 1998 by Netscape Communications Corporation
  - Dedicated to promoting OSS
  - Build bridges among different constituencies in open source community.
  - Educates developers, decision makers and users about the advantages of OSS and OSS related activities
  - Established Open Source Definition (OSD)

  (https://opensource.org/osd-annotated/)

Open Source Definition

- ***Free Redistribution***
  - License shall not restrict any party ***from selling or giving away*** the software as a ***component*** of an aggregate software.
  - License shall not require a royalty or other fee.
- ***Source Code***
  - Program must include ***source code***, and ***allow distribution in source code as well as compiled form.***
  - Where some form of software is not distributed with source code, a well-publicized means of obtaining source code with little reasonable reproduction cost, must be available.
  - Source code must be in ***a human readable format*** and easily understandable with clear annotations

Open Source Definition (cont.)

- ***Derived Works***
  - License must allow modifications and derived works, and allow them to be ***distributed under the same terms as the license of the original soft***ware.
- ***Integrity of the Author's Source Code***
  - License may restrict source-code from being distributed in modified form only if the license allows distribution of "patch files" with the source code for modifying the program.
  - License must permit distribution of software built from modified source code.
  - License may require derived works to carry a different name or version number from the original software.

Open Source Definition (cont.)

- ***No Discrimination Against Persons or Groups***
- ***No Discrimination Against Fields of Endeavor***
- ***Distribution of License***
  - The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of additional license by those parties.
- ***License Must Not Be Specific to a Product***
- ***License Must Not Restrict Other Software Distributed along with the licensed software***
- ***License Must Be Technology-Neutral***

OSS vs. Freeware and Shareware

- ***Freeware*** refers to software available for use at no cost
- ***Shareware*** refers to software provided to users without payment ***on a trial basis and often limited by functionality or time of use***.
- ***OSS vs. Freeware***
  - OSS
  - Freeware / Shareware
    - Developers are holding the software copyrights
    - Proprietary and closed source code
    - License restricts modifications and redistributions of the software

Business Perspectives of OSS

- Most software companies do not disclose their source code and do sell their software without source code
- Some companies are willing to contribute to the OSS. **Why?**
- Large software vendors, like IBM, Dell, HP, and Oracle are also making significant amounts of indirect revenue from their activities with and support of OSS

Major OSS Companies

- Red Hat: Operating system – Linux, http://www.redhat.com/
- Untangle: gateway that blocks spam, spyware, viruses and adware http://www.untangle.com/
- WordPress: Blogging platform, http://wordpress.org/
- OpenBravo: Enterprise resource planning (ERP), http://www.openbravo.com/
- JasperSoft: Business Intelligence (BI) Suite, http://www.jaspersoft.com/
- Canonical: Desktop operating system - Ubuntu Linux http://www.canonical.com/
- SugarCRM: Customer relationship management (CRM), http://www.sugarcrm.com/
- Digium: IP telephony platform, http://www.digium.com/en/
- MySQL: Database, http://www.mysql.com/
- Mozilla Foundation: Web browser, http://www.mozilla.org/en-US/
- Apache Software Foundation: Web server, http://www.apache.org/

Tools for OSS Development

- Communication channels
  - Examples: email, instant messaging, Internet Relay Chat(IRC), web forums, Wikis
- Version control systems
  (http://en.wikipedia.org/wiki/List_of_revision_control_software)
  - Examples: Concurrent Versions System (CVS), Apache Subversion (SVN), Distributed Version Control System (DVCS)
  - Helps manage version control for the files and codes of a project when several developers are working on the project at the same time.
  - Allows several developers to work on the same file at the same time.

Tools for OSS Development (cont.)

- Bug trackers (http://en.wikipedia.org/wiki/Bug_tracking_system)
  - Examples: Bugzilla, Mantis Bug Tracker, Trac, Request tracker, and GNATS
  - Help keep track of the status of various issues (bugs, flaws, security vulnerabilities) in the development of an OSS project
- Testing tools (http://www.opensourcetesting.org/)
  - Examples: Tinderbox, GNU Debugger, XPCOM, Splint
- Package management tools (http://en.wikipedia.org/wiki/List_of_software_package_management_systems)
  - Examples: The Red Hat Package Manager(RPM), Advanced Packaging Tool (APT)
  - Help automate the process of installing, upgrading, configuring, and removing software packages from a computer

Security of OSS

- OSS is ***less secure than Proprietary Software***
  - Difficult to control the quality of software.
  - No responsibility for developers
  - Source code available to attackers
  - Making source code available does not guarantee review
  - A malicious developer can participate in OSS development and intentionally implement malicious functions
  - An attacker may inject virus, worm or malicious code in OSS and redistribute to public
  - OSS often has problems with poor documentation, supporting and patching because there is no enforced control by a single organization
  - Open source development process may not be well defined, and the stages in the development process, such as system testing and documentation may be ignored

http://www.zdnet.com/blog/security/the-empty-debate-over-open-source-security/1623

Security of OSS (cont.)

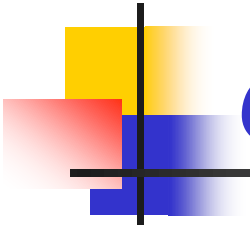
- OSS is ***more secure than Proprietary Software***
  - Commonly used OSS is often more reliable since thousands of independent developers and users test such OSS and fixing vulnerabilities in such OSS.
  - OSS is developed more carefully and clearly due to the fact that OSS will be open to public and reviewed by other developers
  - Due to open communications among software developers of OSS, the software developers have more knowledge for security issues and technologies.

Security of OSS (cont.)

- Some argue OSS is ***more secure*** (cont.)
  - If a user wants to know whether a particular feature is secure, the user can find it by examining the source code
    - Can also find malicious functions, viruses, worms, and backdoors by inspecting the source code
  - Proprietarily software is expected to be more secure because of the secrecy of its source code. However, security through obscurity is not working because keeping vulnerabilities secret does not make the vulnerabilities go away
    - Undiscovered vulnerabilities simply means that the vulnerabilities are time bombs without knowing when they will explode.

Security of OSS (cont.)

- Some argue OSS is ***more secure*** (cont.)
  - For proprietary software, users are forced to accept the ***level of security*** the vendors have chosen to provide. For OSS, users can choose the security level as high as they want by adding more security features.
  - OSS can be developed according to purely technical requirements. It does not require the developers to think about commercial pressure (cost and time-to-market) that often degrades the quality of the software.



OSS or Proprietary Software?

- Which software is more secure? (OSS or proprietary software?)
  - Decide which software you will use carefully based on your business goals, context, budget, and requirements.
  - If you decide to use OSS, you need to plan ahead carefully using it securely

How to Use OSS Securely?

- Choose commonly used OSS
- Identify your security requirements precisely
- Test the OSS thoroughly, review its source code, and verify that the OSS meets your requirements
- Read supporting documents carefully for secure installation, configuration and maintenance.
- Actively engage with various OSS communities
- Add your own security features on OSS
- Sanitize and validate inputs/outputs of OSS