Abstract This essay studies different types of security errors. Specifically, in section 2 we study different security error classification schemes. Examples of security errors based on classifications and their countermeasures are studied in section 3. We see how to understand security errors in computer programs and what are common ways to avoid them.

1 Introduction

There are many descriptions for classifying security schemes. Some examples of them are: RISOS project [5], Protection Analysis project [6], perspective approach [7] and its extension [8] and others. But most of these classification schemes have their downsides: they tend to be ambiguous, making classification difficult and subjects to misclassification. Furthermore, they are rather directed towards research, not to developers working on applications in real life.

Taking these considerations into account, new classification scheme was proposed in [1]. The authors took developer-centric approach to propose a new easily understood classification scheme. This scheme is further studied in section 2.

In addition, we also look at [2] to compare how common security errors in web applications fit into the classification, because it can be considered de facto standard for web application security.

Using public “Common Vulnerabilities and Exposures” database [4] we classify and study some security errors in open source software. The main goal is to study the discovery of errors and their countermeasures. We try to avoid hard technical implementation details while rather studying general principles according to [3], which is a guideline for avoiding security errors defined in the main paper [1].
2 Classification of security errors

In this section, security errors are split into several categories in order of importance. The authors of [1] have also covered additional category ‘environment’, but it will not be covered, because it is rather related to OS environment and configuration. These kinds of errors could be documented by developers and are usually out of their control.

Thus, the seven pernicious kingdoms according to [1] in order of importance are as following.

2.1 Input Validation and Representation

Input validation and representation errors are most common errors found in applications. These errors are caused by trusting user input to be valid; by various metacharacters which break program flow; by different encodings etc.

OWASP Top Ten also includes injection and cross-site script as two main reasons of security errors[2]. They suggest separating all data as trusted and untrusted and have a distinct boundaries for crossing into trusted zone. They also have assessed unvalidated redirects and forwards[3] as the least important security risk. This risk could also be categorized under API abuse, but the risk arises from unvalidated input rather than from redirections.

This kingdom is split to specific cases: buffer overflow; command injection; cross-site scripting; format string; HTTP response splitting; illegal pointer value; integer overflow; log forging; path manipulation; process control; resource injection; setting manipulation; SQL injection; string termination error; unsafe reflection and XML validation.

2.2 API Abuse

API abuses are cases, where different software communicate with each other. It is necessary that all interfaces behaviours are documented and uniform. Failing to do so may lead to security errors. For example, after chroot() syscall, it is required that working directory should be changed with chdir(), because otherwise it is possible to break out of chroot. Furthermore, all open file descriptors have still access outside of chroot. So, obtaining chroot consists of more actions that just one syscall[4].

[4] Linux Programmer’s Manual; chroot(2)
Cases of API abuses are: use of dangerous functions; directory restriction; heap inspection; sockets in J2EE; user identification; exception handling; file system usage; privilege management; strings manipulation and unchecked return value.

### 2.3 Security Features

It is important to make difference between software security and security software. Security software also has to consider authentication, access control, confidentiality, cryptography and privilege management.

OWASP Top Ten lists broken session management † insecure cryptographic storage † failure to restrict URL access † and insufficient transport layer protection † as reasons of security errors. Errors considering security features consists make up the largest part of OWASP Top Ten, so it can be considered that it is most difficult to do correctly in web applications.

This kingdom is split into following categories: insecure randomness; least privilege violation; missing access control; password management; empty password; hard-coded password; password storage; weak cryptography and privacy violation.

### 2.4 Time and State

Modern computers have mostly multiple cores to perform computation. If different threads work on different cores then it may happen that same region of memory is read or written on exactly same time. This leads to memory corruption.

In general distributed programs need to know how to modify and account state. This can be hard to consider because in many cases parallelism is not taken into account.

So, more specific errors regarding to time and state: deadlock; failure to begin a new session upon authentication; file access race condition (TOCTOU); insecure temporary file; threads in J2EE; `System.exit()` in J2EE and signal handling race conditions.

---

5 https://www.owasp.org/index.php/Top_10_2010-A3  
6 https://www.owasp.org/index.php/Top_10_2010-A7  
7 https://www.owasp.org/index.php/Top_10_2010-A8  
8 https://www.owasp.org/index.php/Top_10_2010-A9
2.5 Errors

Error handling is a subset of API usage, but as bugs related to handling errors are so common, the authors dedicated separate kingdom to this subject. Most common mistakes are poor error handling (and no error handling at all) and too descriptive error messages returned to users, as they can be exploited by malicious parties to gather further knowledge about the system.

The categories of this kingdom are: catching `NullPointerException`; empty catch block; overly-broad catch block and overly-broad throws declaration.

2.6 Code Quality

This kingdom covers unexpected behaviour of programs. This is caused by bad programming and leads to poor usability and possible attack channels to malicious user.

OWASP Top Ten considers insecure direct object reference\(^9\) as error in code quality.

This kingdom is divided into: double `free()`; inconsistent implementation; memory leak; null dereference; obsolete code; undefined behaviour; uninitialized variable; unreleased resource and use after free.

2.7 Encapsulation

It is important to have strong isolation between different parts of programs. For example, it should not be possible for unauthenticated user to access some internal data.

The subclasses are: comparing classes by name; data leaking between users; leftover debug code; object hijack; use of inner class; non-final public field; private array-typed field returned from a public method; public data assigned to private array-typed field; system information leak and trust boundary violation.

3 Examples and countermeasures of security errors

We have now covered the classification scheme. For further understanding the motivation of this classification and the use, we will look into some discovered bugs and how they were resolved. For each bug we assign a kingdom and kingdom’s subclass to it. We also look how the bugs were fixed.

\(^9\)https://www.owasp.org/index.php/Top_10_2010-A4
3.1 **CVE-2013-1834**

**CLASSIFICATION:** Encapsulation - data leak between users.

**DESCRIPTION:** *notes/edit.php* in Moodle 1.9.x through 1.9.19, 2.x through 2.1.10, 2.2.x before 2.2.8, 2.3.x before 2.3.5, and 2.4.x before 2.4.2 allows remote authenticated users to reassign notes via a modified (1) userid or (2) courseid field.

The fix unset userid and courseid, thus removing the ability to change other user’s data.

3.2 **CVE-2013-1635**

**CLASSIFICATION:** Input Validation and Representation - path manipulation.

**DESCRIPTION:** PHP does not validate relationship between the *soap.wsdl_cache_dir* directive and the *open_basedir* directive, which allows remote attackers to bypass intended access restrictions by triggering the creation of cached SOAP WSDL files in an arbitrary directory.

Fix for this security error was trivial - implement checking if *soap.wsdl_cache_dir* is confirming to *open_basedir*. In the same commit there is some additional code which checks for terminating characters.

3.3 **CVE-2009-4272**

**CLASSIFICATION:** Time and State - deadlock.

**DESCRIPTION:** A certain Red Hat patch for *net/ipv4/route.c* in the Linux kernel 2.6.18 on Red Hat Enterprise Linux (RHEL) 5 allows remote attackers to cause a denial of service (deadlock) via crafted packets that force collisions in the IPv4 routing hash table, and trigger a routing ”emergency” in which a hash chain is too long.

In a patch which fixed the issue, lock was released before starting emergency hash table rebuild.

3.4 **CVE-2008-5110**

**CLASSIFICATION:** API Abuse - directory restriction.

---

10 [http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-1834](http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-1834)
12 [http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-4272](http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-4272)
13 [http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-5110](http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-5110)
DESCRIPTION: syslog-ng does not call `chdir()` when it calls `chroot()`, which might allow attackers to escape the intended jail.

In the patch, process working directory was changed to one being chrooted into.

3.5 CVE-2012-6063

CLASSIFICATION: Code Quality - double `free()`

DESCRIPTION: Double free vulnerability in the `sftp.mkdir()` function in `sftp.c` in `libssh` before 0.5.3 allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via unspecified vectors.

After receiving errors from called functions, request was freed and then code execution was jumped to error section, where it was again freed. Contrary to CERT recommendation on using `goto` in C, memory freeing was removed from error section.

3.6 CVE-2010-2772


DESCRIPTION: Siemens Simatic WinCC and PCS 7 SCADA system uses a hard-coded password, which allows local users to access a back-end database and gain privileges, as demonstrated in the wild in July 2010 by the Stuxnet worm.

As the software is close-sourced, it is not known if they have patched the bug later, but eventual recommendation from Siemens was not to change default password (“2WSXcder”) as it would “break the system” and instead it was recommended to guard the systems with other tools.

4 Conclusion

We have seen from the examples, that this classification schema has made easy to classify security errors and it is unambiguous. The naming of kingdoms and specific examples aids the auditors to evaluate errors.

Based on CVE database, currently there are no classification schemas used to classify bugs in the database. We strongly suggest that new assigned CVE identifiers should also consider the classification. Having bugs already pre-classified helps project managers in assigning tasks to members. This leads to more efficient work flow and more rapid responses to discovered bugs.

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-6063
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-2772
Even though there is extensive example collection at [3], it does not provide any suggestions on fixing the bugs. These examples should also be provided to help developers fix bugs in a secure fashion as there have been cases when fixing one bug introduces another one.

When referencing to OWASP Top Ten[2], we saw that most of web application security errors are covered under input validation and security features. This can be accounted to fact that web applications require more user interaction than dedicated software and thus introduce more entry points for possible errors. Furthermore, there are some security errors which are not covered by main seven pernicious kingdoms[1], but rather by additional environment kingdom. This implies that configuration plays greater part in web applications than in dedicated software.

References