Two threat patterns that exploit “Security misconfiguration” and “Sensitive data exposure” vulnerabilities

ROHINI SULATYCKI, Florida Atlantic University
EDUARDO B. FERNANDEZ, Florida Atlantic University

We present threat patterns that describe attacks against applications that take advantage of security misconfigurations in the application stack and applications that expose sensitive data. These patterns provide insight on how to build and configure applications that might contain sensitive data and are also helpful in evaluating the security of existing applications.

Categories and Subject Descriptors: D.2.11 **[Software Engineering]** Software Architecture – Patterns; D.5.1 D.4.6 **[Security]**

General Terms: Design

Key Words and Phrases: Misuse patterns, security patterns, application security, vulnerabilities, exploits, misconfiguration, and sensitive data

1. INTRODUCTION

To design a secure application, we first need to understand possible threats to the application. For this purpose we introduced the concept of misuse patterns (Fernandez et al., 2007), which describe how an information misuse is performed. Later, we proposed *threat patterns* (Uzunov and Fernandez, 2013), to describe the steps of an attack leading to several related misuses. Both patterns describe how an attack is performed from the point of view of the attacker. They define the environment where the attack is performed, countermeasures to stop it, and provide forensic information in order to trace the attack once it happens. A security defense misconfiguration is a *vulnerability,* taking advantage of this vulnerability is a threat (potential attack) which can lead to reading unauthorized information (a *misuse*). In particular, threat patterns are useful for developers because once they determine that a possible attack can happen in the environment, the pattern will indicate what security mechanisms are needed as countermeasures. Also, threat patterns can be very useful for forensic examiners to find useful evidence information after the attack has been performed. Finally, they can be used to evaluate an existing system and verify if it can handle specific threats. Note that a threat pattern describes a complete attack, e.g. stealing information from a database, not just specific steps used to perform the attack, such as SQL injection or buffer overflow (both can be used in the same threat or individually in many threats). Threat patterns take advantage of specific vulnerabilities and can be described with respect to the corresponding vulnerability or they can be defined with respect to a set of vulnerabilities that allow the attack to proceed.

The Open Web Application Security Project (OWASP) publishes a list of the ten most critical web application security risks called the OWASP Top 10 (OWA10). The OWASP Top 10 is a widely adopted document for application security and has been developed with broad consensus from security experts worldwide. However, their descriptions of common threats do not emphasize architectural aspects and they mix vulnerabilities with threats. We are converting the OWASP descriptions into threat patterns because we believe they will be more useful in this way (the affected architecture is very important); we have already written patterns for two of these threats: “Compromising applications using components with known

Author's address: R. Sulatycki, 3 Sylvan Way, Parsippany, New Jersey 07054; email: sulatyckir@dnb.com; E.B.Fernandez, Department of Computer and Electrical Engineering and Computer Science. Florida Atlantic University 777 Glades Rd. Boca Raton, FL 33431; email: fernande@fau.edu
 ACM 978-1-4503-3847-9/15/07...$15.00
DOI: http://dx.doi.org/10.1145/2855321.2855368

vulnerabilities" and "Direct access to objects using uncontrolled references" (Sulatycki and Fernandez 2015a).

The Internet has been developing very rapidly during the last decade. New technologies are being introduced at a rapid pace and applications are being built using new and emerging technologies. Applications are composed of a variety of components including in-house developed components, outsourced components, COTS components, web services, and mashups. They are deployed using a complex mix of technologies including web servers, application servers, and database servers. Securing the application stack is a difficult endeavor and any security misconfiguration within the application stack can lead to an application compromise.

A number of applications contain sensitive data such as user credentials, financial data, credit cards and personal information. Often, sensitive data does not get discarded and is stored permanently. Failure to protect sensitive data can lead to data exposure, which can have severe legal and financial impacts for application owners.

We present here two threat patterns to describe attacks based on security misconfiguration and sensitive data exposure. Our audience includes architects, system designers, and web application developers. Section 2 presents the template we use to describe threat patterns (Fernandez, 2013); it is based on the POSA template (Buschmann et al., 1996) with tailoring of some sections. In Section 3, we present a threat pattern for application deployment, Security Misconfiguration. In Section 4 we present a threat pattern that exploits Sensitive Data Exposure. In Section 5, we present some discussion on how threat patterns can be used, and we offer some conclusions and possible future work.

2. TEMPLATE FOR MISUSE/ THREAT PATTERNS

2.1 Name

The name of the pattern should correspond to the generic name given to the specific type of threat in standard attack repositories.

2.2 Intent or thumbnail description

A short description of the intended purpose of the pattern (what problem it solves for an attacker).

2.3 Context

It describes the generic environment including the conditions under which the attack may occur. This may include minimal defenses present in the system as well as standard vulnerabilities of the system.

2.4 Problem for the attacker

From an attacker's perspective, the problem is how to find a way to attack the system. The *forces* indicate what factors may be required in order to accomplish the attack and in what way; for example, which vulnerabilities can be exploited.

2.5 Solution

This section describes the solution of the attacker's problem, i.e., how the attack can reach its objectives and the expected results of the attack. UML class diagrams show the system units involved in the attack. Sequence or collaboration diagrams show the exchange of messages needed to accomplish the attack.

2.6 Affected system components (Where to look for evidence)

The pattern should represent, typically using a class diagram, all components that are important to prevent the attack. From a forensic viewpoint, it describes what information can be obtained at each stage tracing back the attack and what can be deduced from this data.

2.7 Known uses

Specific incidents where this attack occurred are preferred but for new vulnerabilities, where an attack has not yet occurred, specific scenarios for the potential attack are enough.

2.8 Consequences for the attacker

Discusses the benefits and drawbacks of a misuse pattern from the attacker's viewpoint. The enumeration includes good and bad aspects and should match the forces.

2.9 Countermeasures

It describes the security measures necessary in order to stop, mitigate, or trace this type of attack. This implies an enumeration of which security patterns are effective against this attack.

2.10 Related Patterns

Discusses other threat patterns with different objectives but performed in a similar way or with similar objectives but performed in a different way.

2.9 Related Patterns

Discusses other misuse patterns with different objectives but performed in a similar way or with similar objectives but performed in a different way.

3. TAKING ADVANTAGE OF SECURITY MISCONFIGURATION

3.1 Intent

An attacker may take advantage of misconfigurations such as unpatched software, unnecessary additional features such as application server administrative console or enabled default accounts, to compromise the application

3.2 Context

Applications are deployed using a complex application stack: web servers, application servers, database servers, third party components and custom code. In many cases there are multiple teams that are responsible for the security of different components: database administrators, system administrators, development teams etc. These diverse teams need to work together to ensure the security of the overall system. In this environment it is easy for a security misconfiguration to occur at any level of the application stack.

3.3 Problem

To perform some type of misuse it is necessary to identify some vulnerability within the application stack.

These attacks can be performed by taking advantage of the following vulnerabilities:

- Default or weak passwords are enabled within the application, which can be used to compromise systems contained within the applications stack. In June 2013, US Cert (US-CERT 2013) issued an alert on the risks of default passwords being used on the Internet.
- An application stack might contain unpatched software. Exploit code for the vulnerability might be publicly available, be developed or purchased from underground channels.

The attack is facilitated by:

- There are a number of dictionaries such as CVE (CVE) that publish vulnerabilities and related exploits such as vulnerabilities in specific versions of databases and frameworks. They may include misconfiguration exploits.
- The emergence of vulnerability markets (Böhme, R. 2005) provides an economic incentive for researchers to search for and to disclose information on vulnerabilities. These include vulnerability disclosures related to security misconfigurations.
- The existence of bug bounty programs provides an economic incentive to search for and to disclose information on vulnerabilities. Additionally, vulnerability disclosure is seen as a status symbol by many security experts.

3.4 Solution

An attacker can identify vulnerabilities within the application stack due to misconfiguration and misuse these vulnerabilities to compromise the application, the file system, and/or other systems.

3.4.1 Structure

Figure 1 shows a class diagram for compromising applications using security misconfigurations. There are three types of servers: **Database**, **Web**, and **Application**. Applications can be **Custom** or **Administrative.** The Attacker operates on the **Application.** An application may have different **Configurations.** Users have **Accounts** to use the Application.

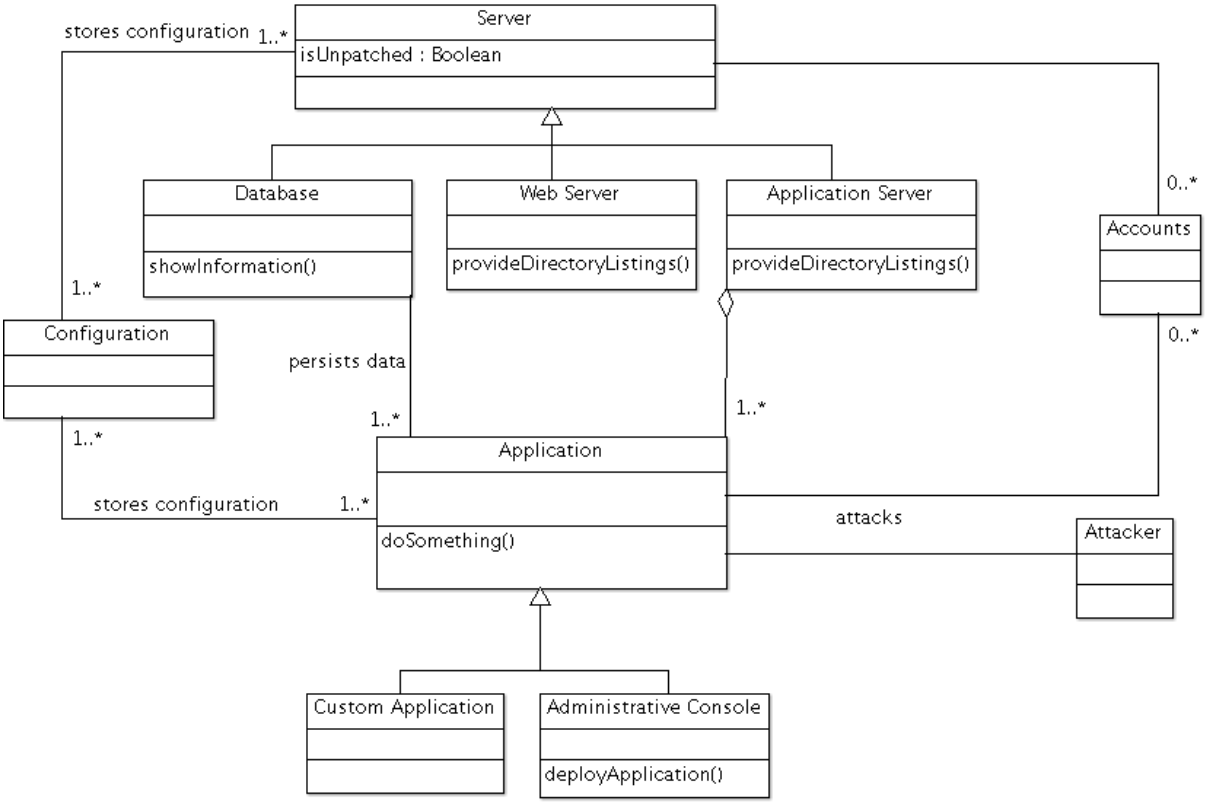

Figure 1. Class Diagram for Compromising Applications Security Misconfigurations Misuse Pattern

3.4.2 Dynamics

The following use case describes a threat that take advantage of security misconfigurations.

UC1: Compromising an application with security misconfigurations (Fig. 2)
Summary: The Attacker compromises an application whose security is misconfigured.

Actor: Attacker

Precondition: The Attacker must have access to the application. The application has misconfigured its security. The security misconfiguration must be accessible to the attacker.

Description:

- The attacker scans the application and finds the security misconfiguration.
- The attacker then misuses the security misconfiguration to compromise the application.

Postcondition:
The application is compromised, which can lead to a variety of misuses (see Section 3.6).

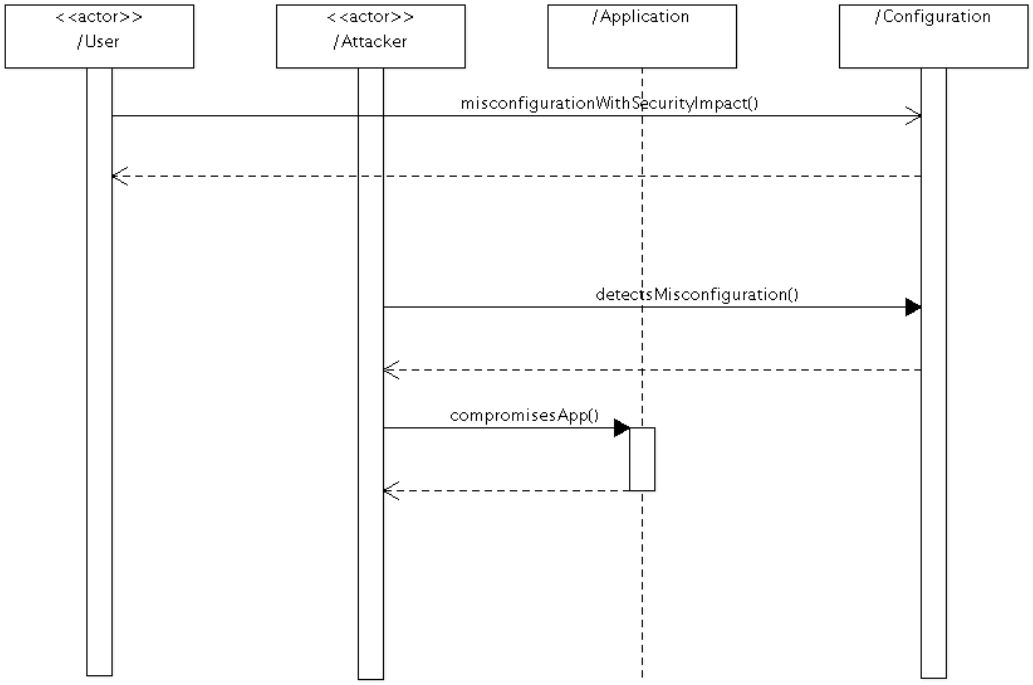

Figure 2. Sequence Diagram for the use case *Compromising an application with security misconfigurations*

3.5 Known uses (incidents)

Some incidents where these patterns (apparently) were used are:

- A report issued in November 2014 on the security of the Office of Personnel Management (OPM) systems stated found that the agency did not perform regular security scans of its systems (NYTimes); OPM keeps records and security clearance information for millions of current and retired federal employees. In June 2015, a massive breach at OPM of over 4 million personnel records was revealed.
- Anthem, the US second health insurer had a data breach in 2014 (SHRM, US-CERT). As many as 80 million customers and the company's employees have had their employment data, addresses, Social Security numbers, and birth dates stolen. Anthem's database was accessed after logon information for database administrators had been compromised.
- Target suffered a data leak in 2013 (NYTimes). Target said at least 40 million credit cards were compromised by the breach during the holiday shopping season, and the attack might have resulted in the theft of personal information, such as email addresses and telephone numbers, from as many as 110 million people. The attack was enabled by a lack of security in point of sale terminals.
- In April 2011, Sony suffered a massive data breach due to the insecurity of its game network (Wikipedia).

The descriptions of these attacks do not indicate which specific vulnerabilities were exploited. For example the following misconfiguration could cause a vulnerability that an attacker can exploit: https://www.samba.org/samba/security/CVE-2009-2813.html. If a user in /etc/passwd is misconfigured to have an empty home directory (::) and the automated [homes] share is enabled, or an explicit share is created with that username, then any client connecting to that share name will be able to access the whole filesystem from root (/) on downwards, subject to local file system permissions applied to the connecting user. It does not properly handle errors in resolving pathnames, which allows remote authenticated users to bypass intended sharing restrictions, and read, create, or modify files, in certain circumstances involving user accounts that lack home directories.

3.6 Consequences

Some of the benefits of the threat pattern are the following:

- An attacker can get control of the compromised system to obtain some confidential information e.g. credentials, financial information or modify confidential information.
- An attacker might be able to disable or delete the application using the administrative console leading to a denial of service condition.
- The attacker might be able to expand the attack to other systems and obtain more confidential information.
- The attacker might be able to perform modification of the application stack for financial gain, e.g. , to insert ransomware.

Possible sources of failure for the attacker include:

- The application may have defenses against the intended attacks.
- The exploit code may not work.
- The infected system might be sufficiently sandboxed to prevent the attacker from accessing other systems. However, in this case the application itself might still be compromised.

3.7 Countermeasures and Forensics

Security misconfiguration can be stopped by the following countermeasures:

- Provide a repeatable hardening (e.g., remove default and test accounts) process to lock down the system configuration.
- Apply software updates and patches in a timely manner.
- Provide a system to monitor the security of all components of the application stack in public databases, mailing lists, and security mailing lists, and keep them up to date.
- Require scans and security testing of the entire application stack.
- Use security patterns in the application stack. Design applications using appropriate security methodologies (Uzunov et al. 2012).

Either the application itself or the administrative console would have traces of the attack.

3.8 Related Patterns

- Misuse patterns for cloud computing: Malicious virtual machine creation is possible due to poor security configuration [K. Hashizume et al. 2011].

- Many applications utilize and process sensitive data as part of the application functionality. Financial applications contain credit card data and other financial data while healthcare applications contain patient- related information. Most applications contain user credentials and personally identifiable information (PII). Many applications do not protect this data appropriately and it can be accessed through insecure components.

4. GAINING ACCESS TO SENSITIVE DATA

4.1 Intent

Many applications utilize and process sensitive data as part of the application functionality. Financial applications contain credit card data and other financial data while healthcare applications contain patient-related information. Most applications contain user credentials and personally identifiable information (PII). Many applications do not protect this data appropriately (or at all) and an attacker can access it with little work.

4.2 Context

Applications processing sensitive data are accessible to users. Often sensitive data is not appropriately discarded and kept in backups and in databases for long periods of time. If applications do not protect sensitive data an attacker may gain access to this data and exploit this information in various ways, such as ID theft, blackmail, or financial theft.

4.3 Problem

To perform some types of misuse it is necessary to identify an application that processes sensitive data.

The attack can be performed by taking advantage of the following vulnerabilities:

- In an application sensitive data can usually be obtained from any of these locations: client side caches, memory locations, while in transit, or while at rest. Any weakness in securing any of these locations could allow an attacker to obtain sensitive information.
- In January 2014, US Cert (US-CERT 2014) issued an alert on the risks of malware targeting POS systems. The risks include memory scraping.

4.4 Solution

When a user publishes an application users are able to access the application. An attacker can identify vulnerabilities within the application communication, caching directives, encryption or persistence mechanism to get unauthorized access to sensitive data.

4.5 Variants

- An application sends sensitive data over insecure communication channels. The attacker establishes a man in the middle position to monitor and intercept traffic sent between the application and an end user and obtain the sensitive data.
- An application uses improper cache directives on pages that contain sensitive data. The user accesses sensitive application data using a browser. Since the application has not set proper caching directives the data gets cached in the users browser. The attacker gains access to users computer and accesses the sensitive data.
- Due to a directory-listing flaw, an attacker is able to find a directory containing archives. The archives are not protected by the application's security controls and the users is able to download the archives and access sensitive information.
- Due to a SQL Injection attack, an attacker is able to obtain arbitrary data from the applications database. The data is not encrypted securely and the attacker is able to decrypt the data and obtain sensitive information.

4.5.1 Structure

Figure 3 shows a class diagram for the Sensitive Data Exposure misuse pattern. **Applications** can be **Database** or **Archive** applications. Applications are accessed from a **Browser**, which has a **Cache** and interacts with the application through a **Proxy Server.**

4.5.2 Dynamics

UC1: Compromise an Application that exposes sensitive data (Fig. 4)
Summary: The Attacker compromises an application exposing sensitive data.

Actor: Attacker

Precondition: The application must be available to the attacker.

Description:

- The attacker finds that the application does not protect sensitive data securely while in transit or at rest.
- The data is then exposed to the attacker.

Postcondition:

The attacker accesses unauthorized sensitive information. Possible misuses are described in Section 4.7.

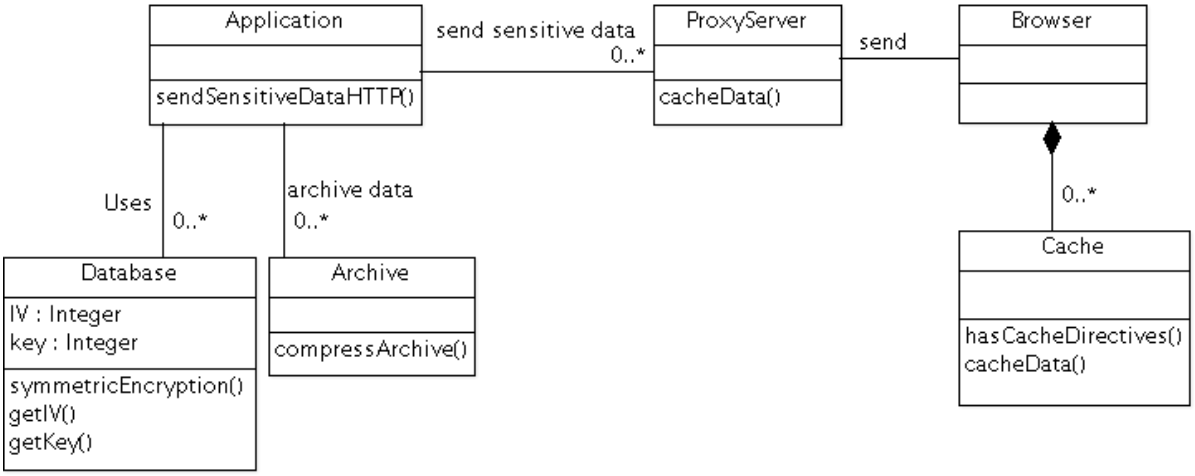

Figure 3. Class Diagram of the Sensitive Data Exposure Misuse Pattern

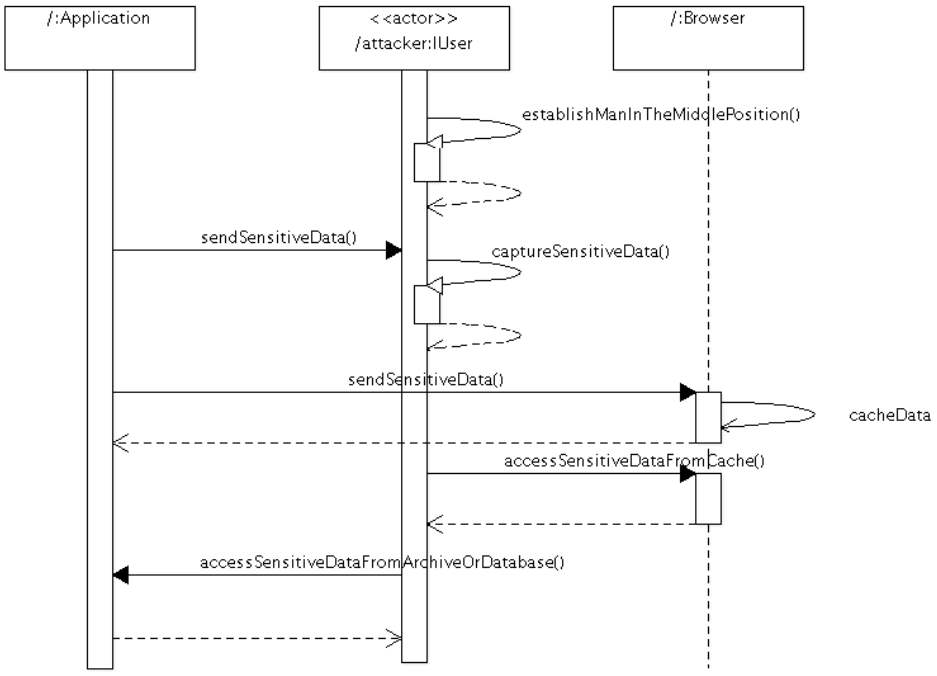

Figure 4. Sequence Diagram for the use case *Compromise an Application that exposes sensitive data*

4.6 Known uses

Some of the known uses for the misuse pattern are:

- Target suffered a data leak in 2013 (NYTimes). Target said at least 40 million credit cards were compromised by the breach during the holiday shopping season, and the attack might have resulted in the theft of personal information, such as email addresses and telephone numbers, from as many as 110 million people. Reuters (Reuters) has reported that memory scraping software was believed to have been used to steal credit card information in this breach. The credit card data was reported to have been unencrypted in memory locations of point-of-sale (POS) devices.

- Since the data theft at Target became public, several other retailers have acknowledged breaches of their own, including Neiman Marcus and the arts and crafts store Michaels. Those two breaches are believed to have come from the same Eastern European group that attacked Target, and may have attacked as many as six other retailers.

- Anthem, the US second health insurer had a data breach in 2014 (SHRM). As many as 80 million customers and the company's employees have had their employment data, addresses, Social Security numbers, and birth dates stolen. Anthem's database was accessed after logon information for database administrators had been compromised, so this is also an incident for security misconfiguration.

- The video games publisher Ubisoft suffered a data breach in 2013. Maybe 58 million people had user names, email addresses and encrypted passwords accessed (BBC News).

4.7 Consequences

Some of the benefits of the misuse pattern are the following:

- The attacker can obtain sensitive data contained while in transit on the network.
- The attacker can gain sensitive data from browser caches.
- The attacker can obtain sensitive data contained in directory listings.
- The attacker can obtain sensitive data from the application database.

Possible sources of failure for the attacker include:

- The data might be encrypted within the browser caches.
- The data might be encrypted while in transit.
- The data might be encrypted in archives.
- The database might be enforcing secure encryption and the attacker might not be able to decrypt the sensitive data.

4.8 Countermeasures and Forensics

Compromising sensitive data exposure can be stopped by the following countermeasures:

- Implement proper authorization controls (Fernandez13) to ensure that the user is authorized for the requested object including static content.
- Implement strong encryption and enforce secure key management.
- Discard data as soon as it is not needed.
- Implement secure caching for all sensitive data
- Enforce that all sensitive data be sent over a secure channel.
- Require security testing of all applications before being made available to users.
- Design applications using appropriate security methodologies (Uzunov et al., 2012).

Where can we find evidence of this attack?

- Application, Database, and/or OS logs.

- We can audit a compromised application.
- Network monitoring logs

4.9 Related Patterns

- Taking advantage of security misconfigurations: An attacker may take advantage of misconfigurations such as unpatched software, unnecessary additional features such as application server administrative console or enabled default accounts, to access unprotected or weakly-protected sensitive data.

- Authorization: Describe who is authorized to access specific resources in a system, in an environment in which we have resources whose access needs to be controlled. It indicates for each active entity, which resources it can access, and what it can do with them. This pattern can prevent access to data anywhere in the application stack.

5. DISCUSSION AND CONCLUSIONS

Designers need to first understand possible threats before designing secure systems. However, identifying threats is not enough; we need to understand how a whole misuse is performed by taking advantage of them. *Threat* and *misuse* patterns appear to be a good tool to understand how misuses are performed. It is possible to build a relatively complete catalog of threat and misuse patterns for application security. Having such a catalog we can analyze a specific application and evaluate its degree of resistance to these misuses and we are in the process of building this catalog. The architecture (existing or under construction) must have a way to prevent or at least mitigate all the threats that apply to it. When potential customers use an application they must have assurance on what threat/misuses the application is able to prevent. Many providers do not want to show their security architectures; showing their list of threat/misuse patterns would give them a way to prove a degree of resistance to misuses without having to show their security details.

We illustrated our ideas with two specific patterns. We are continuing developing misuse patterns for application security in order to create a relatively complete catalog for it that can be used by application developers. Finally, we intend to incorporate these patterns into a secure systems design methodology.

ACKNOWLEDGMENTS

Our shepherd, Kristian Beckers, provided valuable comments that significantly improved this paper. The workshop participants at EuroPLoP provided useful suggestions.

REFERENCES

BBC News , Ubisoft warns millions of video gamers of hack attack
http://www.bbc.com/news/technology-23159997, July 3rd, 2013

CVE, Common Vulnerabilities and Exposures, https://cve.mitre.org/

R. Böhme, Vulnerability markets. What is the economic value of a zero-day exploit? Proceedings of 22nd Chaos Communication Congress, (Berlin, Germany, dec. 27-30, 2005).

F. Braz, E.B.Fernandez, and M. VanHilst, "Eliciting security requirements through misuse activities" Procs. of the 2nd Int. Workshop on Secure Systems Methodologies using Patterns (SPattern'07). 2008.

E.B. Fernandez, J.C. Pelaez, and M.M. Larrondo-Petrie, "Attack patterns: A new forensic and design tool", Procs. of the Third Annual IFIP WG 11.9 Int. Conf. on Digital Forensics, Orlando, FL, Jan. 29-31, 2007.
Chapter 24 in Advances in Digital Forensics III, P. Craiger and S. Shenoi (Eds.), Springer/IFIP, 2007, 345-357.

E.B. Fernandez, N. Yoshioka, and H. Washizaki, "Modeling misuse patterns", 4th Int. Workshop on Dependability Aspects of Data Warehousing and Mining Applications (DAWAM 2009), in conjunction with the 4th Int.Conf. on Availability, Reliability, and Security (ARES 2009). March 16-19, 2009, Fukuoka, Japan

E.B.Fernandez, “Security patterns in practice: Building secure architectures using software patterns”. Wiley Series on Software Design Patterns. 2013

K Hashizume, E. B. Fernandez, and N. Yoshioka, "Misuse patterns for cloud computing: Malicious virtual machine creation"", Procs. of the Twenty-Third International Conference on Software Engineering and Knowledge Engineering (SEKE 2011), Miami Beach, USA, July 7-9, 2011

The New York Times, http://www.nytimes.com/2014/03/14/business/target-missed-signs-of-a-data-breach.html

The New York Times, http://www.nytimes.com/2015/06/06/us/chinese-hackers-may-be-behind-anthem-premera-attacks.html?_r=0

Reuters, http://www.reuters.com/article/2014/01/12/us-target-databreach-retailers-idUSBREA0B01720140112

OWASP, “The Ten Most Critical Web Application Security Risks.” 2010. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

SHRM, http://www.shrm.org/hrdisciplines/technology/articles/pages/lessons-learned-from-anthem-data-breach.aspx#sthash.yjYXEIRD.dpuf

Wikipedia, http://en.wikipedia.org/wiki/2011_PlayStation_Network_outage

Rohini Sulatycki and E.B. Fernandez, “Two threat patterns: “Compromising applications using components with known vulnerabilities” and “Direct access to objects using uncontrolled references”, Procs. of 4th AsianPLoP (Pattern Languages of Programs) 2015, Tokyo, Japan, March 2015.

US-CERT, https://www.us-cert.gov/ncas/current-activity

US-CERT, https://www.us-cert.gov/ncas/alerts/TA13-175A

US-CERT, https://www.us-cert.gov/ncas/alerts/TA14-002A

A. V.Uzunov and E.B.Fernandez, “An Extensible Pattern-based Library and Taxonomy of Security Threats for Distributed Systems”- Special Issue on Security in Information Systems of the *Journal of Computer Standards & Interfaces*. 2013. http://dx.doi.org/10.1016/j.csi.2013.12.008

A. V. Uzunov, E.B.Fernandez, and K. Falkner, “Engineering Security into Distributed Systems: A Survey of Methodologies”, Journal of Universal Computer Science, Vol. 18, No. 20, 2012, pp. 2920-3006.
http://www.jucs.org/jucs_18_20/engineering_security_into_distributed

http://dl.acm.org/citation.cfm?id=1233393&dl=ACM&coll=DL&CFID=576080885&CFTOKEN=56360445#URLTOKEN#
http://dl.acm.org/citation.cfm?id=2721986&dl=ACM&coll=DL&CFID=576080885&CFTOKEN=56360445#URLTOKEN#
http://dl.acm.org/citation.cfm?id=2721974&dl=ACM&coll=DL&CFID=576080885&CFTOKEN=56360445#URLTOKEN#
http://dl.acm.org/citation.cfm?id=500084&dl=ACM&coll=DL&CFID=576080885&CFTOKEN=56360445#URLTOKEN#

http://dl.acm.org/author_page.cfm?id=81100383974&CFID=576080885&CFTOKEN=56360445