Proceedings of the XXVI International Symposium on Nuclear Electronics & Computing (NEC'2017)
Becici, Budva, Montenegro, September 25 - 29, 2017

INFORMATION DISCLOSURE IN SOFTWARE INTENSIVE SYSTEMS

V.T. Dimitrov[a]

Faculty of Mathematics and Informatics, University of Sofia, 5 James Bourchier Blvd., Sofia, 1164, Bulgaria

E-mail: [a] cht@fmi.uni-sofia.bg

Research and investigations on computer security problems show that the most malicious problem is the information disclosure. Today this problem is enormous in the context of the new cloud services. The paper is an overview of the main computer security components: attacks, vulnerabilities and weaknesses with a focus on the last ones. An approach to information disclosure weaknesses formalization and its usage for automated weakness' discovery are discussed.

Keywords: data anonymity, information disclosure, computer security

1. Introduction

The main terms used in computer security are attack, vulnerability and weakness.

Attack is a sequence of actions (manual, automatic or both) initiated by an attacker against a software intensive system to gain an unauthorized access to or some benefits from the system. There is always motivation for the attack. Attack targets are software weaknesses.

Software weakness is an intentional or unintentional bug in the system architecture, design, implementation or configuration.

Vulnerability is a weakness that can be successfully exploited by an attack. Not every weakness available in the software is a vulnerability - it can be protected by some security mechanisms.

The weaknesses represent the computer security statics of the system. The attacks represent computer security dynamic of the system. The vulnerabilities are bridge between the static and dynamic.

This conceptual system (attack, vulnerability and weakness) is supported in many computer security repositories.

"An information exposure is the intentional or unintentional disclosure of information to an actor that is not explicitly authorized to have access to that information." [1]

Alternative terms are information leak and information disclosure. The term "information leak" in computer security is used additionally in the sense of resource leak, i.e. improper tracking of resources which can lead to exhaustion. The term "information disclosure" does not refer to disclosure of security-relevant information. It used mainly in vulnerability databases and policies and legal documents.

Unauthorized information disclosure implies that the information is sensitive in some way, i.e. it is classified as such one by the corresponding authorities. For the purpose of this paper it is not important how the information is classified as sensitive.

Data masking is a fundamental component of data security used for data anonymization (de-identification). It enables organizations to de-identify, mask and transform sensitive data. Anonymized data can be used for research purpose. A range of transformation techniques can be applied to substitute sensitive data with contextually-accurate but fictionalized data to produce accurate research results. By masking personally-identifying information, organizations can protect the privacy and security of confidential data, and support compliance with local and international privacy regulations.

There are 3 types of data anonymization: masking identifiers in unstructured data, privacy preserving data analyses (interactive scenario) and transforming structured data (non-interactive scenario). [2]

The next aspects in anonymization algorithms must be balanced: use cases, types of data, risk and thread models, privacy models (syntactic or semantic), transformation methods and utility measures (loss of information).

Important aspects of the use cases are:

- Who or what processes the data and in which way? Humans (types of analyses, interactive or non-interactive) or machines (classification, clustering).
- How will the data be released? Access control (open or restricted) or continuous data publishing (multiple views, re-release – incremental or new attributes).
- Is the data distributed? Collaborative environments (vertical or horizontal).

Important properties of the data are data type (relational or transactional), data dimensionality, and data clusterization. Transactional data consists of set-valued attributes.

Among the transformation techniques are string literal values, character substrings, random or sequential numbers, arithmetic expressions, concatenated expressions, date aging, lookup values and intelligence.

2. Motivations

Anonymization must be treated in the context of the concrete use cases. Data anonymization is data protection from unauthorized access, i.e. there is an intended or unintended attack for data disclosure. From that point of view, data disclosure is a vulnerability that occurs as result of an attack on some weakness. Therefore data anonymization is a mitigation or prevention of data disclosure weaknesses in software intensive systems.

The main subject for data anonymization are so called Personally Identifiable Information (PII) and Protected Health Information (PHI).

PII is information that alone or in combination with other information allows a person to be identified. This is information that permits individual persons to be revealed from the mass of information. Part of the PII is information associated with the person.

PII can be financial (card number, bank account number and balance on it), employment details (salary, position occupation), personal (photo, biometric data, birth date, sex, marital status), education (college, university, qualifications), contacts (e-mail, phone number), medical data (past and current diseases).

PHI is collected, generated, stored and transmitted by the health care vendors. This information directly or indirectly identifies the individuals.

Some of the reasons for data anonymizations:

1. The business generates sensitive data and these data must be protected.
2. The malicious use of personal data is subject of regulatory and legal prosecutions.
3. The misuse of sensitive data generate enormous loses for the business.
4. There are operational risks in the context of outsourcing and cooperation.
5. There are legal and compliance requirements.

The main goal of the current research is to identify the weaknesses that can gain to data disclosure. For that purpose Common Weaknesses Enumeration (CWE) [3] is used.

3. Weaknesses

"CWE-200: Information Exposure" is a class weakness. It status is incomplete. It participates in many views and is starting point for this research. CWE-200 is the ancestor of the next weaknesses:

- CWE-201: Information Exposure Through Sent Data (variant) (draft)
- CWE-202: Exposure of Sensitive Data Through Data Queries (variant) (draft)
- CWE-203: Information Exposure Through Discrepancy (class) (incomplete)
  - CWE-204: Response Discrepancy Information Exposure (base) (incomplete)
  - CWE-205: Information Exposure Through Behavioral Discrepancy (base) (incomplete)
    - CWE-206: Information Exposure of Internal State Through Behavioral Inconsistency (variant) (incomplete)
    - CWE-207: Information Exposure Through an External Behavioral Inconsistency (variant) (draft)
  - CWE-208: Information Exposure Through Timing Discrepancy (base) (incomplete)
- CWE-209: Information Exposure Through an Error Message (base) (draft)
  - CWE-210: Information Exposure Through Self-generated Error Message (base) (draft)
    - CWE-535: Information Exposure Through Shell Error Message (variant) (incomplete)
    - CWE-536: Information Exposure Through Servlet Runtime Error Message (variant) (incomplete)
    - CWE-537: Information Exposure Through Java Runtime Error Message (variant) (incomplete)

- CWE-211: Information Exposure Through Externally-generated Error Message (base) (incomplete)
- CWE-550: Information Exposure Through Server Error Message (variant) (incomplete)

- CWE-212: Improper Cross-boundary Removal of Sensitive Data (base) (incomplete)
- CWE-213: Intentional Information Exposure (base) (draft)
- CWE-214: Information Exposure Through Process Environment (variant) (incomplete)
- CWE-215: Information Exposure Through Debug Information (variant) (draft)
  - CWE-11: ASP.NET Misconfiguration: Creating Debug Binary (variant) (draft)
- CWE-226: Sensitive Information Uncleared Before Release (base) (draft)
  - CWE-244: Improper Clearing of Heap Memory Before Release ('Heap Inspection') (variant) (draft)
- CWE-359: Exposure of Private Information ('Privacy Violation') (class) (incomplete)
  - CWE-202: Exposure of Sensitive Data Through Data Queries (variant) (draft)
- CWE-497: Exposure of System Data to an Unauthorized Control Sphere (variant) (incomplete)
- CWE-524: Information Exposure Through Caching (variant) (incomplete)
  - CWE-525: Information Exposure Through Browser Caching (variant) (incomplete)
- CWE-526: Information Exposure Through Environmental Variables (variant) (incomplete)
- CWE-538: File and Directory Information Exposure (base) (draft)
  - CWE-527: Exposure of CVS Repository to an Unauthorized Control Sphere (variant) (draft)
  - CWE-528: Exposure of Core Dump File to an Unauthorized Control Sphere (variant) (draft)
  - CWE-529: Exposure of Access Control List Files to an Unauthorized Control Sphere (variant) (incomplete)
  - CWE-530: Exposure of Backup File to an Unauthorized Control Sphere (variant) (incomplete)
  - CWE-532: Information Exposure Through Log Files (variant) (incomplete)
    - CWE-533: Information Exposure Through Server Log Files (variant) (incomplete)
    - CWE-534: Information Exposure Through Debug Log Files (variant) (draft)
    - CWE-542: Information Exposure Through Cleanup Log Files (variant) (incomplete)
  - CWE-539: Information Exposure Through Persistent Cookies (variant) (incomplete)
  - CWE-540: Information Exposure Through Source Code (variant) (incomplete)
    - CWE-531: Information Exposure Through Test Code (variant) (incomplete)
    - CWE-541: Information Exposure Through Include Source Code (variant) (incomplete)
    - CWE-615: Information Exposure Through Comments (variant) (incomplete)
  - CWE-548: Information Exposure Through Directory Listing (variant) (draft)
  - CWE-651: Information Exposure Through WSDL File (variant) (incomplete)

- CWE-598: Information Exposure Through Query Strings in GET Request (variant) (draft)
- CWE-612: Information Exposure Through Indexing of Private Data (variant) (draft)

Now, let's analyze this hierarchy of weaknesses.

Every CWE node can be in one of the next statuses:

- "stable" means that the node is ready for public discussion.
- "draft" means that only an initial description is available and further clarification must be done.
- "incomplete" means that the node is not carefully investigated in details.

The life cycle of the node is draft-incomplete-stable. In the last state, the node can be fully utilized.

First, all these weaknesses are in status "incomplete" or "draft". In that case, how useful is the presented information for further research? It is clear that with information in such a state is impossible to be done more formal research. The hierarchy is under development and only after one or two new versions would be in status dominated by "stable" states.

Second, the hierarchy is organized in very strange way. Today, object-oriented approach dominates in the programming and computer professionals can expect that hierarchy is organized following the logic class-base-variant, i.e. the most specific are variants that can be generalized to bases and the last one can be generalized to classes. On the top are views - in that case the Research View.

Several words about views, classes, bases and variants:

- The view organizes weaknesses in a hierarchy from specific point of view.
- The class describes a set of weaknesses with common characteristics. It description usually is independent of specific programming language or technology.
- The base is an abstract description more specific than the class, but it can be used for weakness detection and prevention.
- The variant has the most detailed description linked with some programming language or technology.

CWE-200 hierarchy contains nodes of the same type in relationship parent-child. For example CWE-532 (variant) has as children the variants CWE-533, CWE-534 and CWE-542.

In that hierarchy nodes of intermediate types are not obligatory. For Example, CWE-201 (variant) directly is a child of CWE-200 (class).

The generalization, in object-oriented programming and design, uses a single concept to generalize more specific elements in one more abstract one. In the CWE case, this means that several variants have to be generalized in one base, and that several bases have to be generalized in one class because they are at different abstraction levels. But from the CWE-200 hierarchy it is clear that the concepts of class, base and variant are not used for that purpose. For example CWE-200 have as children several variants (CWE-201 etc.), a class (CWE-203), and several bases (CWE-209 etc.).

The concepts of class, base and variant are used as node attribute describing its abstraction level and they have no relation with hierarchy organization.

Now, the question is "How CWE-200 hierarchy is organized?" After careful investigation of the contents it becomes clear that the hierarchy is organized following attack vectors hierarchy.

An attack vector is a path or means by which an attacker can gain access to the system. Attack vectors enable attackers to exploit system vulnerabilities.

Attack vectors at the top of CWE-200 hierarchy are:

- data transmission (CWE-201);
- query execution (CWE-202);
- discrepancies (CWE-203):
  - response discrepancies (CWE-204);
  - behavior discrepancies (CWE-205):
    - internal state behavioral inconsistency (CWE-206);
    - external state behavioral inconsistency (CWE-207).
  - timing discrepancies (CWE-208).

- error messages (CWE-209);
- etc.

Third, CWE-202 participates two times in the hierarchy: one time as a child of CWE-200 and the second time as a child of CWE-359. This hierarchy is at least strange.

A hierarchy of weaknesses must help to detect them in the code. It must be a base for further research and investigation on these weaknesses. This means that the hierarchy must be organized in object-oriented style - from more abstract to more specific elements. Every node in this hierarchy must be self-contained and useful, i.e. its formal specification must not be trivial one. If this does not happened - the node contains pointless description. The node specification must be the base for automatic detection of the corresponding weakness. The description must not be a common parlance on the topic.

4. Mitigations

Information disclosure can be established during the system architecture, design or implementation phases. It is not related with specific programming languages or technologies. This class of weaknesses more frequently can be found for the architecture paradigm of mobile applications.

Consequences from the successful exploitation of information disclosure weaknesses are on confidentiality because the attackers can read application data.

This class of weaknesses can be detected using the following methods:

- automatic static analyses of binary code or bytecode with partial efficiency;
- dynamic analyses with automatic interpretation of the results with high efficiency;
- dynamic analyses with manual interpretation of the results with partial efficiency;
- manual static analyses of source code with high efficiency;
- automatic static analyses of source code with high efficiency;
- architecture or design review with high efficiency.

This class of weaknesses can be mitigated during architecture / design phase using the principle of privilege separation. The system must be partitioned in safe areas of trust. Sensitive data must not cross safe area boundaries. The system must be built on these safe areas. The privileges must be managed by the principle of least privilege, i.e. the trusted entity should receive only the privilege needed to perform its operations and when the need of such a privilege is way it must be dropped.

5. Acknowledgement

Presented in this paper results are part of the project "Methods for Data Analysis and Knowledge Discovery in Big Sequencing Datasets" (supported by the National Science Fund of Bulgaria, Contract I02/7/2014).

6. Conclusion

Data anonymization is the perfect solution for data protection, because even in the case of successful attack, the attacker access useless data, but there are two problems with data anonymization:

1. How useful for data analysis are anonymized data?
2. The process for data anonymization still remains a subject for data disclosure attacks. Therefore, it must be protected enough for that kind of attacks using above mentioned weaknesses.

The idea to use software weaknesses for data security improvement is a good idea - a preventive action. It is possible to prevent weaknesses as early as possible in the software life cycle. Later on, to remove discovered vulnerabilities is expensive or even impossible.

On the other hand, hierarchy of information disclosure weaknesses is currently under development - there are no one weakness in stable state. This hierarchy is only informative one and as a whole not usable. Will the hierarchy be usable when it reaches the stable state is under question that would be discussed in another place.

It is clear that to protect data from disclosure using CWE weaknesses would not happened for now. Then what to do? An answer is to use available repositories for attacks and vulnerabilities. Every important technology or product has such repositories.

There are no links among these specific repositories. It is possible a vulnerability reported in one repository (and even currently removed) to be available for the same kind of technology or product (and even not reported). Typical example is CVE-2002-2031 that has been reported for Internet Explorer and is not available for the newest versions, but is in full power and even not reported for all other widespread browsers.

References

[1] MITRE Corp., CWE-200: Information Exposure, http://cwe.mitre.org/data/definitions/200.html (visited on 07/30/2017).

[2] ARX, ARX – Powerful Data Anonymization, http://arx.deidentifier.org (visited on 07/30/2017).

[3] 3. MITRE Corp., Common Weaknesses Enumeration, http://cwe.mitre.org (visited on 07/30/2017).