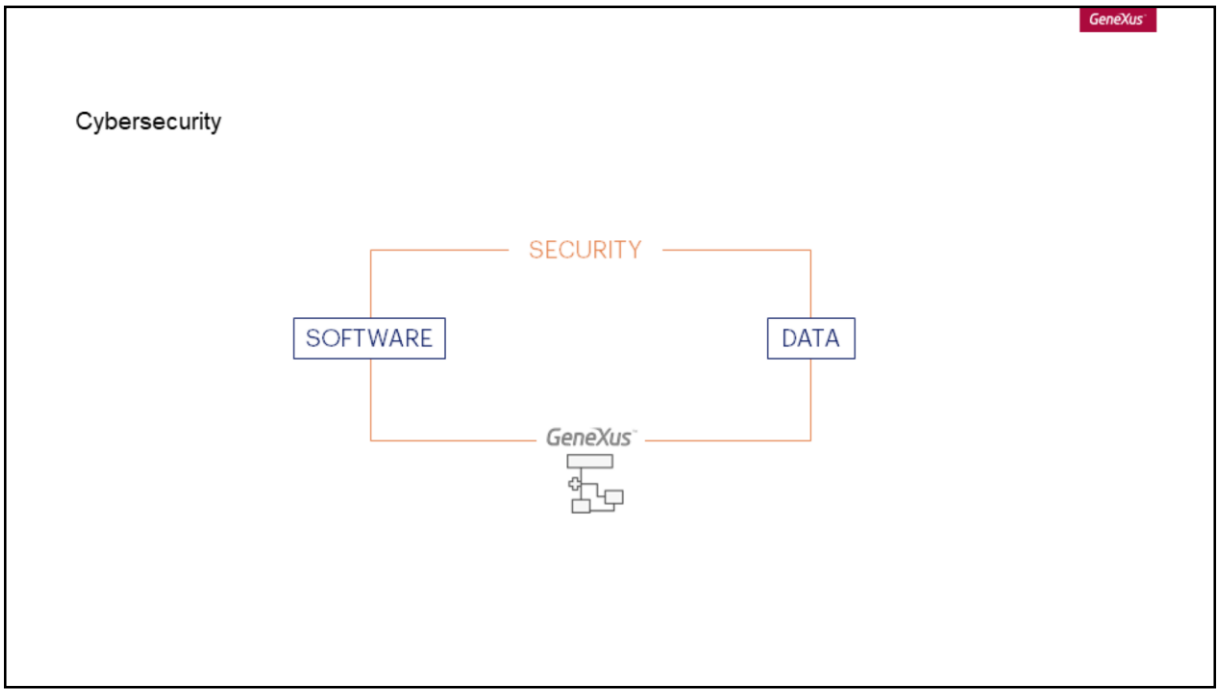
Security in GeneXus

GeneXus 16

GeneXus

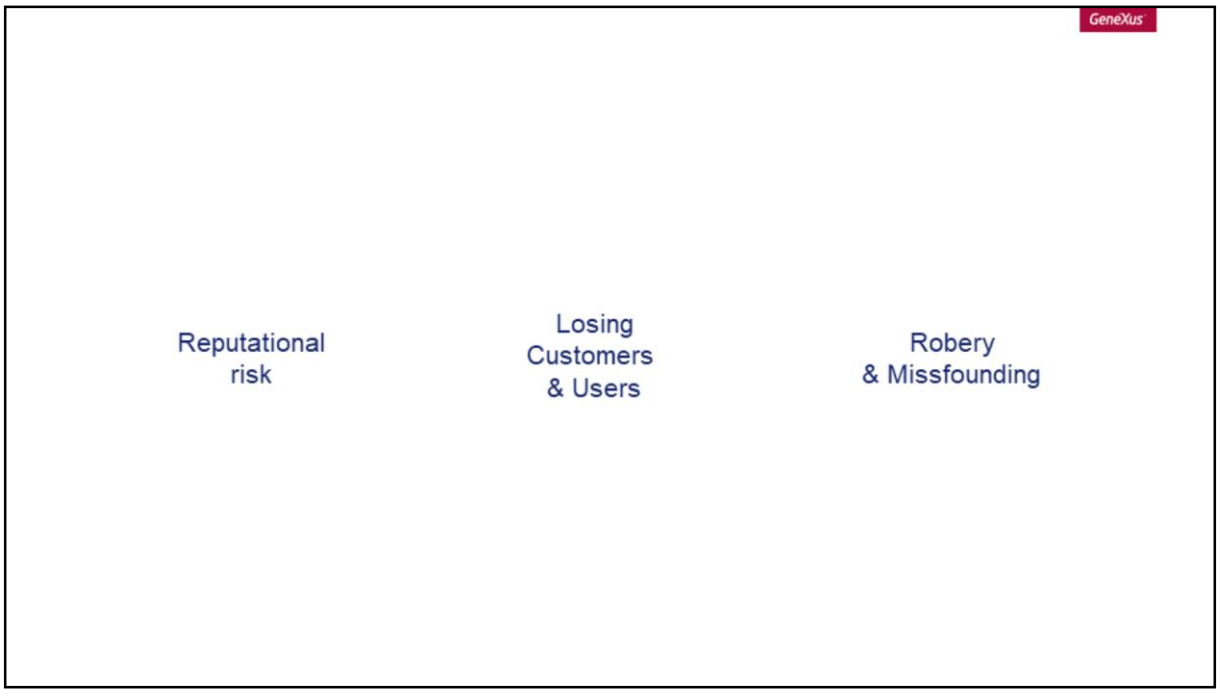
CYBERSECURITY



Computer security (also called cybersecurity) focuses on protecting software infrastructure, and more specifically on data protection. This discipline includes the design of methods, standards and techniques that aid in developing secure and reliable systems.

Cybersecurity relates to different layers within systems: network, servers, databases, and applications, among others. Each layer implies the existence of threats and countermeasures. Applications developed with GeneXus are no exception to that.

System security is important:



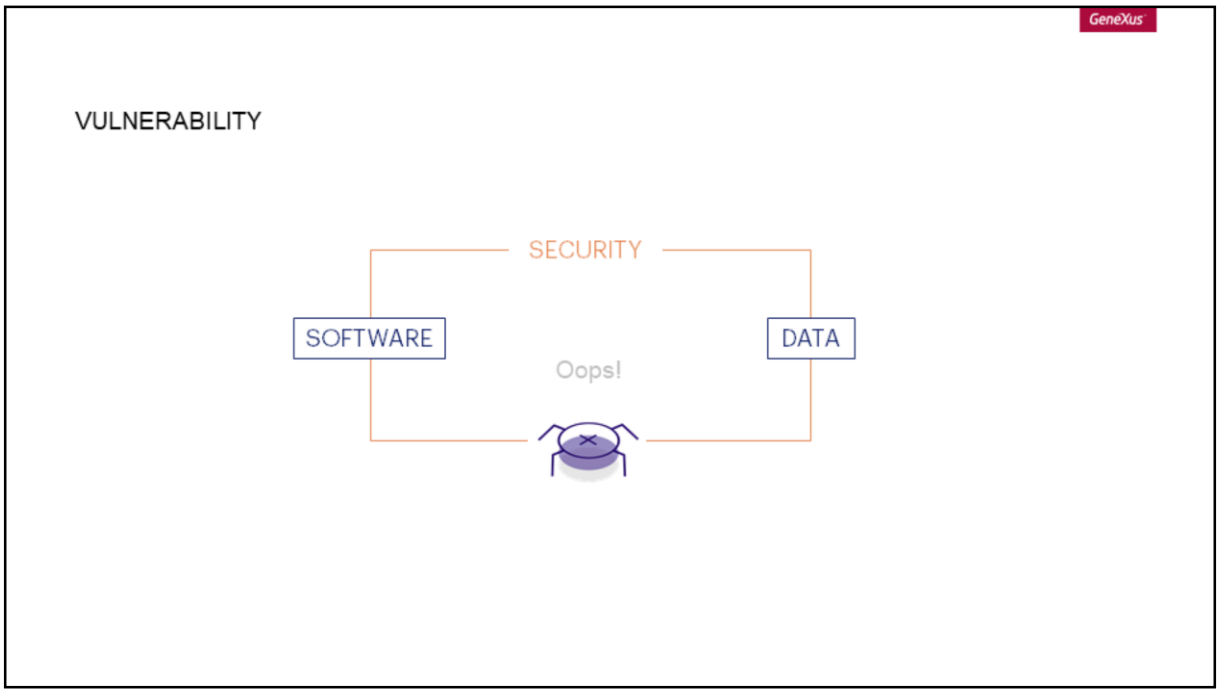
The consequences of security gaps range from an organization's reputational risk that will result in users and customers lost when non-authorized information is made public, to money and goods stolen from the organization and demands from users involved, in addition to fines from different state agencies.

New and stricter regulations relative to privacy, data protection and security gap publications are currently in the making. One example, amidst a world where the abuse of computer systems is in the rise, is the European General Data Protection Regulation –EU GDPR.

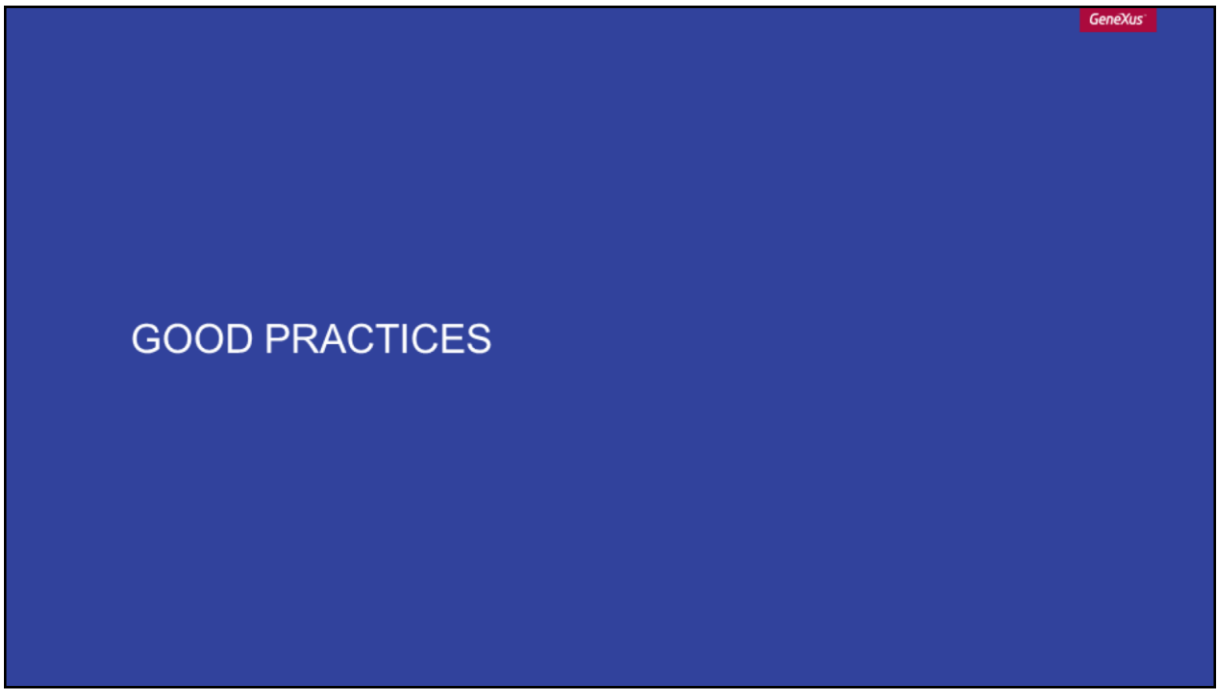
http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/

GeneXus

VULNERABILITY

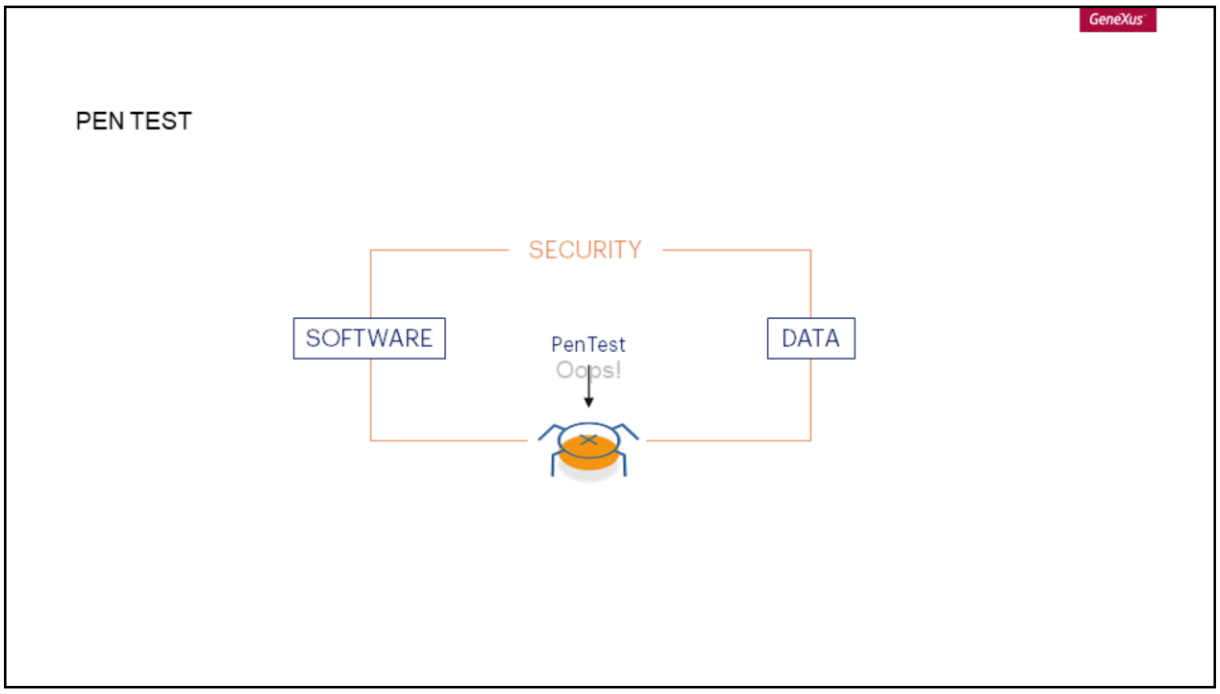


A **vulnerability** implies a flaw or a bug in the system that any agent might benefit from to gain non-authorized access the system, to either steal information or incur in abuse of the resources so that the system becomes impaired to function properly.

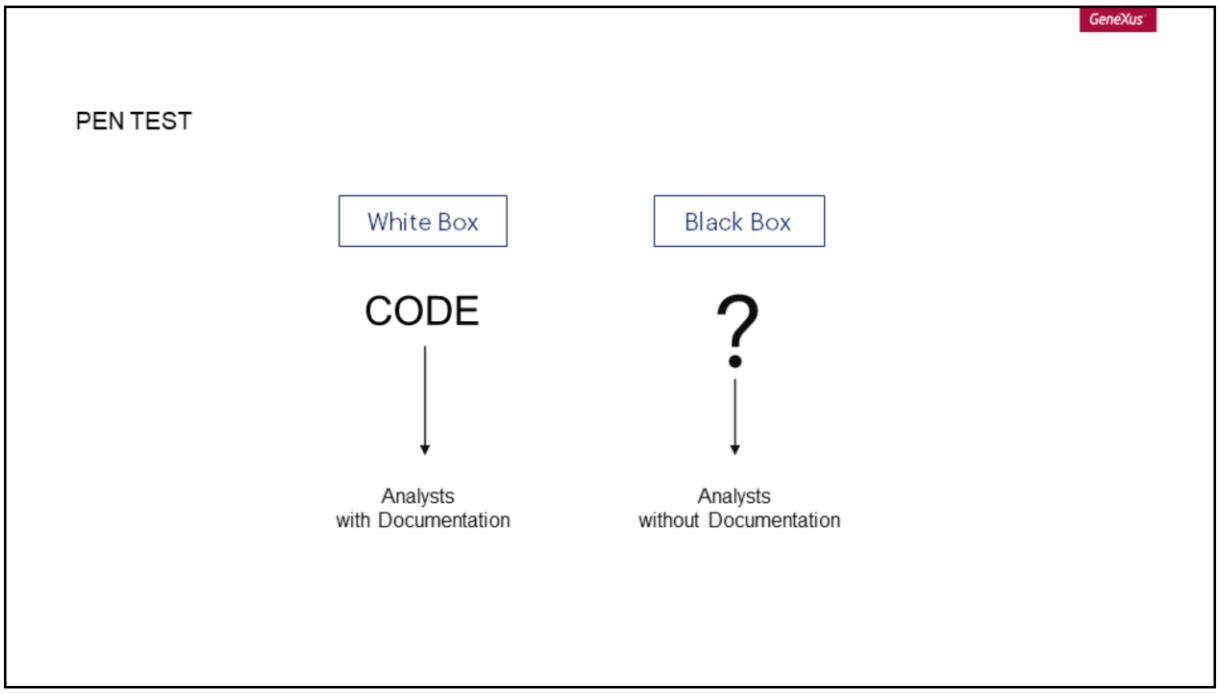


Using best programming practices makes it possible to reduce such risks,

as suggested in the Open Web Application Security Project. This includes all guidelines, manuals and procedures for codification that are available on the Web.

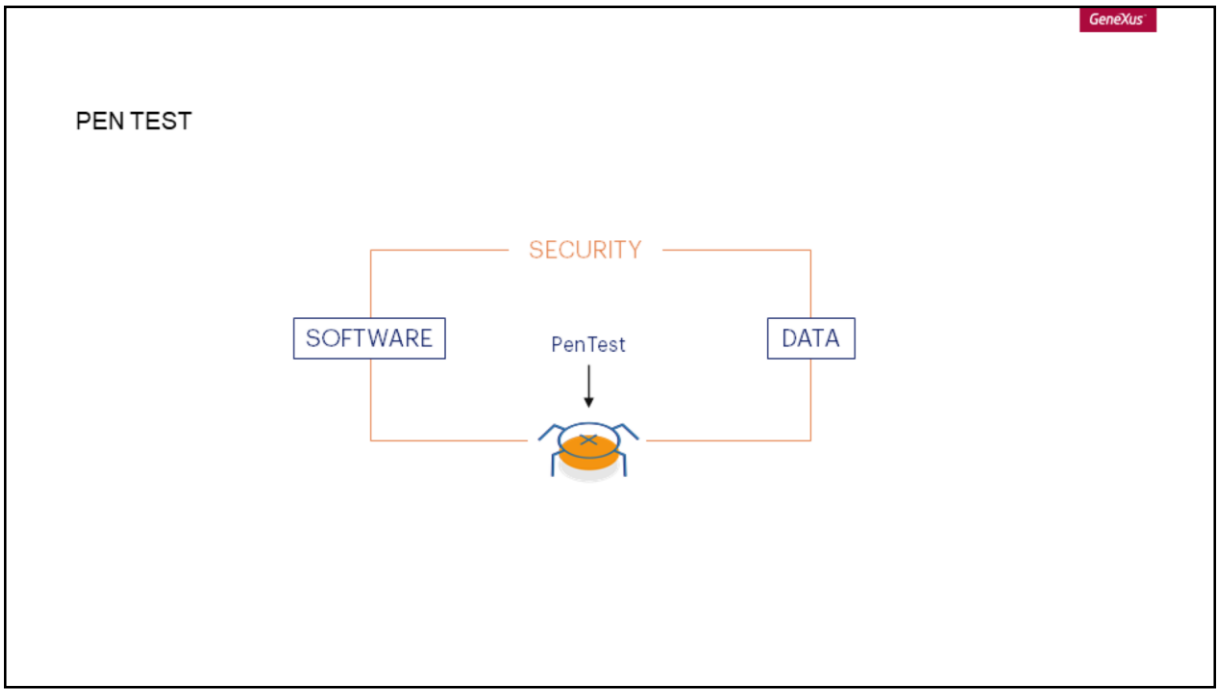


. In this regard, the better-known instance in the secure development cycle is the “pen test”, which represents a simulated attack that has been previously authorized to assess the system’s security and to identify vulnerabilities that were not anticipated during the development stage.



Pen tests may be white box or black box, depending on whether the analyst has the system's code and documents available or not.

To such end, it is common to use automated analysis tools applied to static or dynamic code, whose validity is limited to the interpretation of the security analyst and to the context of the system analyzed.



. It is important to point out that, despite being the most widely known instance of a safe development cycle, the pen test is not the only instance in the cycle. The correction of vulnerabilities detected in this stage is more costly than the case of vulnerabilities detected or prevented in earlier stages.

However, a successful pen test with satisfactory results does not guarantee that a system will not have any vulnerability, just as a functional test is no guarantee that a system is totally bug-free.

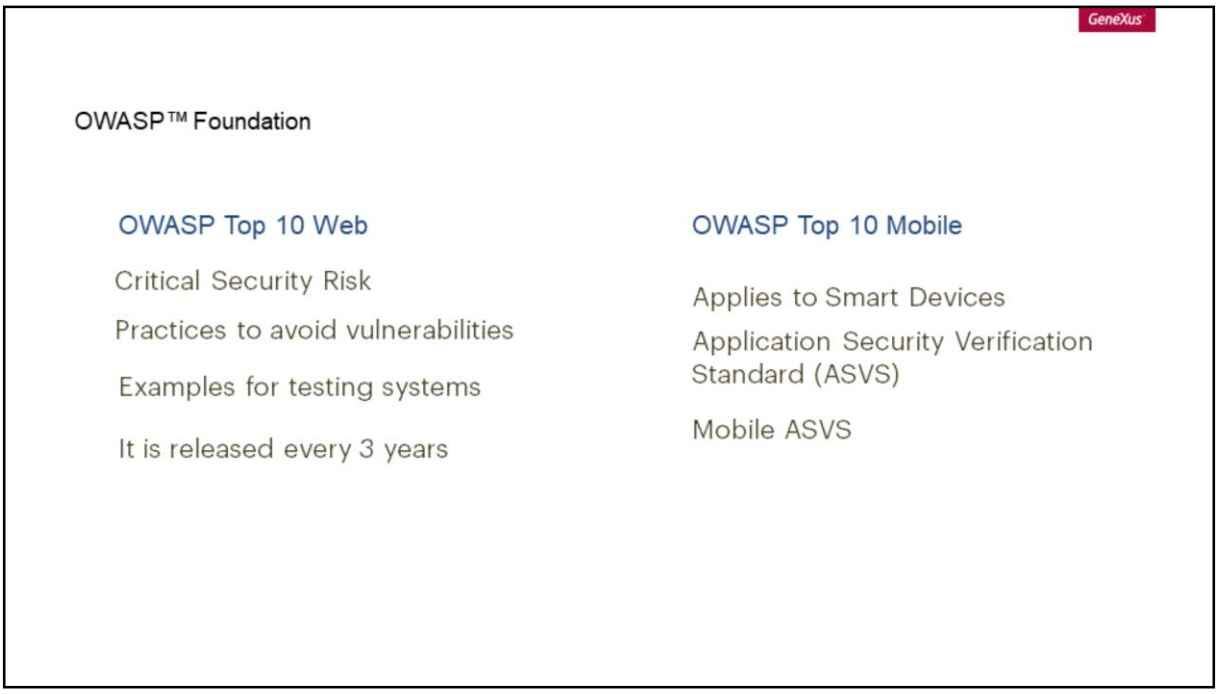
The Open Web Application Security Project is an agnostic technology community, open and non-profit, that facilitates the development, acquisition and maintenance of reliable apps for organizations.

GeneXus

OWASP™ Foundation

Standards information

Good practices



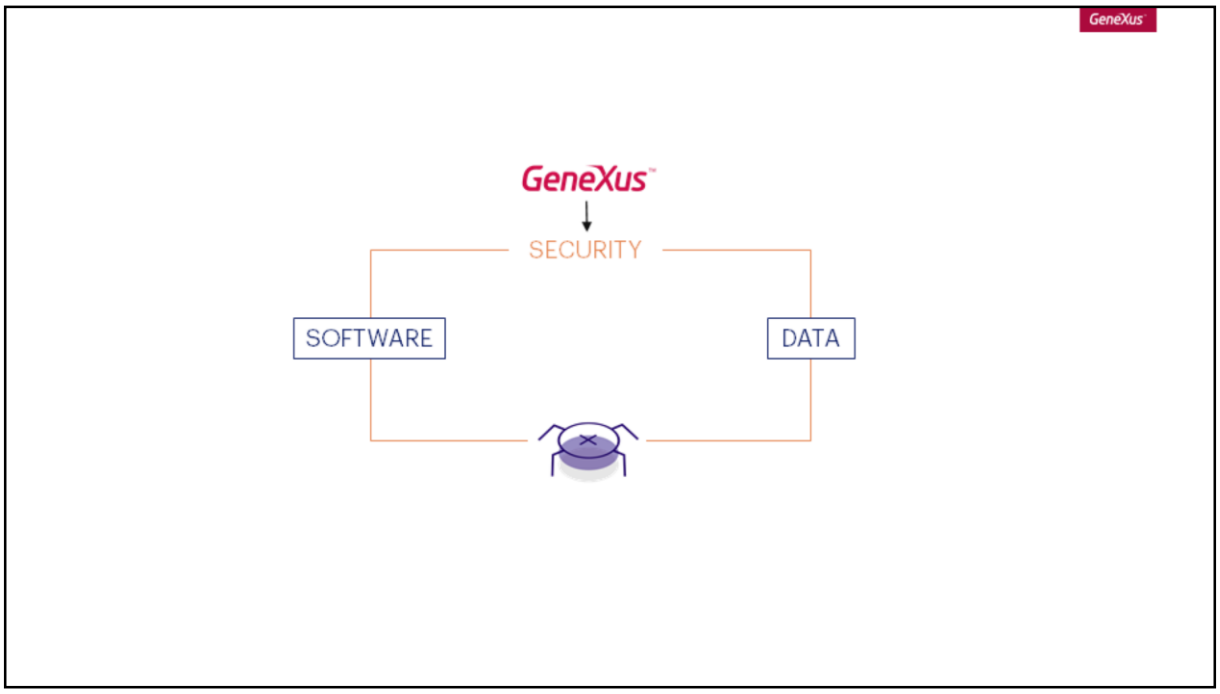
The Open Web Application Security Project contains, in turn, several projects. One of the main and most significant ones is the **Top 10 Web** project, which comprises a document representing the most critical security risks –considering impact and frequency– about which there is consensus among experts.

The purpose of the document is to generate awareness about the issue of system security and to provide guidelines regarding best practices in order to avoid the vulnerabilities listed, as well as to give examples for testing systems that might be subjected to such vulnerabilities and recommend automated testing tools for the necessary analysis.

The Top 10 Web is released every three years, with the one corresponding to the year 2017 being the latest release now.

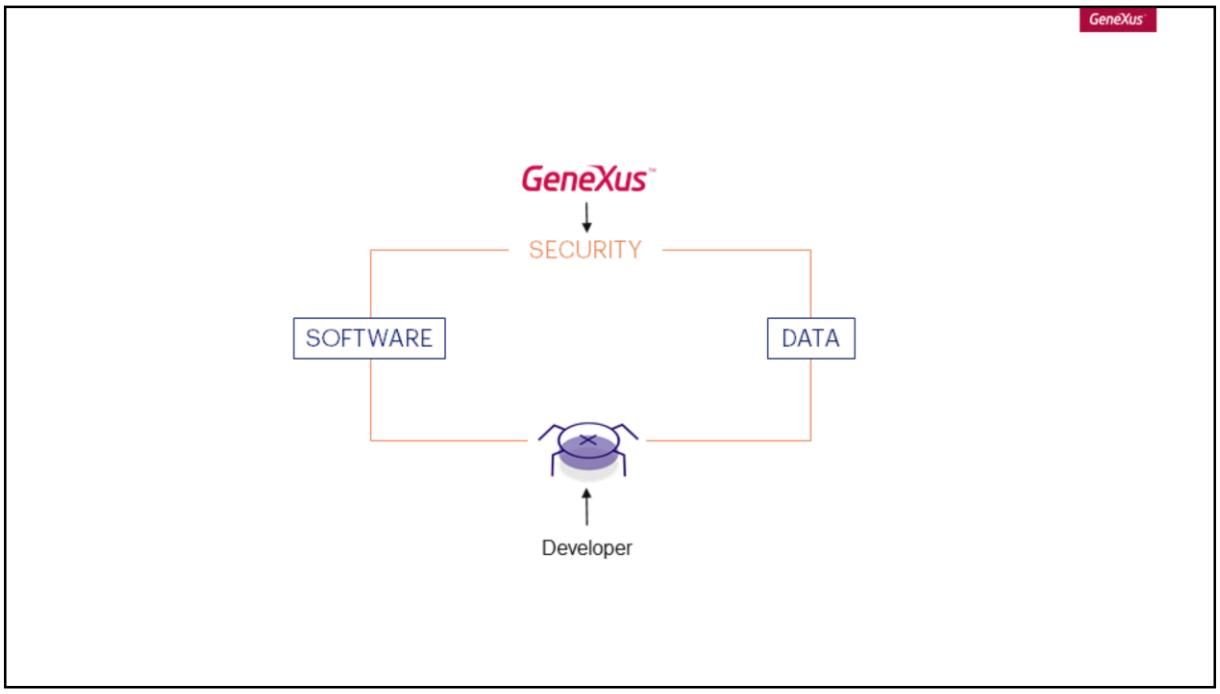
There is also the **Top 10 Mobile**, whose latest release is the 2016 version, applicable to Smart Devices.

The Open Web Application Security Project also includes a project, called *ASVS,* which defines verification levels, also understood as minimum-security requirements that are necessary depending on the application's context. Additionally, there is **Mobile ASVS**, whose current version (as of 2018) is 1.1.



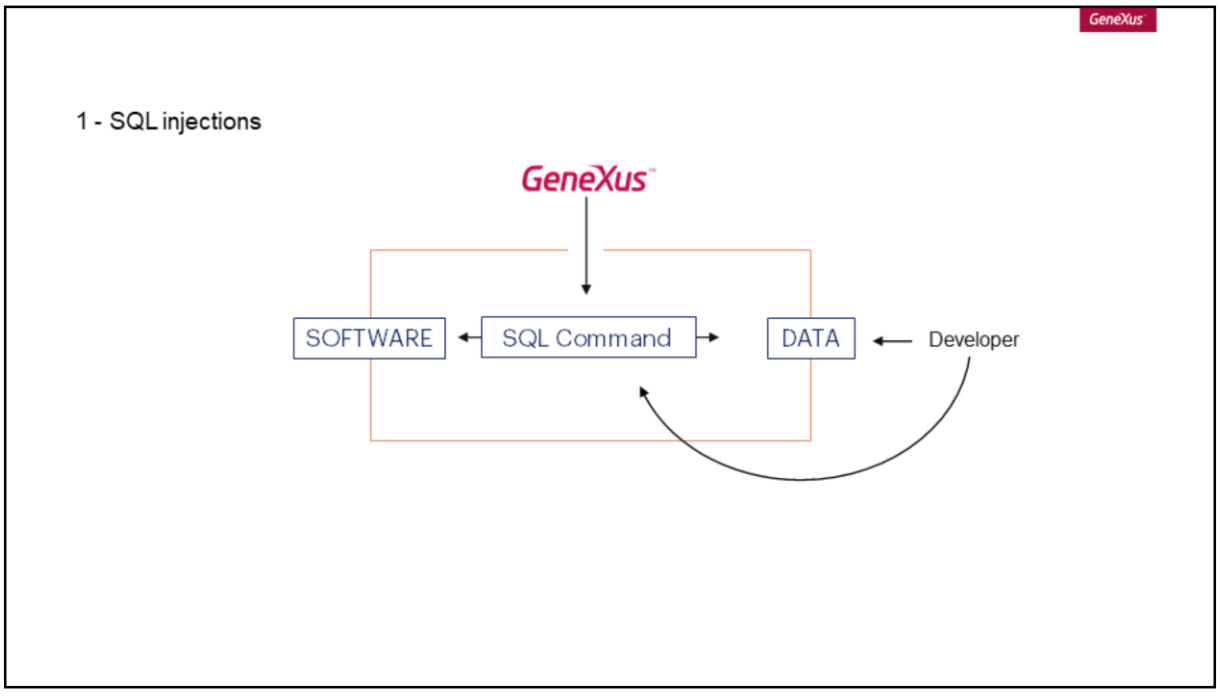
So, what about GeneXus and security controls?

GeneXus automatically implements the security controls that may be inferred from the knowledge base. In other words, vulnerabilities originating in codification errors are mitigated when a language like GeneXus is used.



Nevertheless, when it comes to using the tool, the developer might introduce vulnerabilities or security defects that could result in flaws and errors, at the business logic level as well as in relation to the GeneXus procedural code.

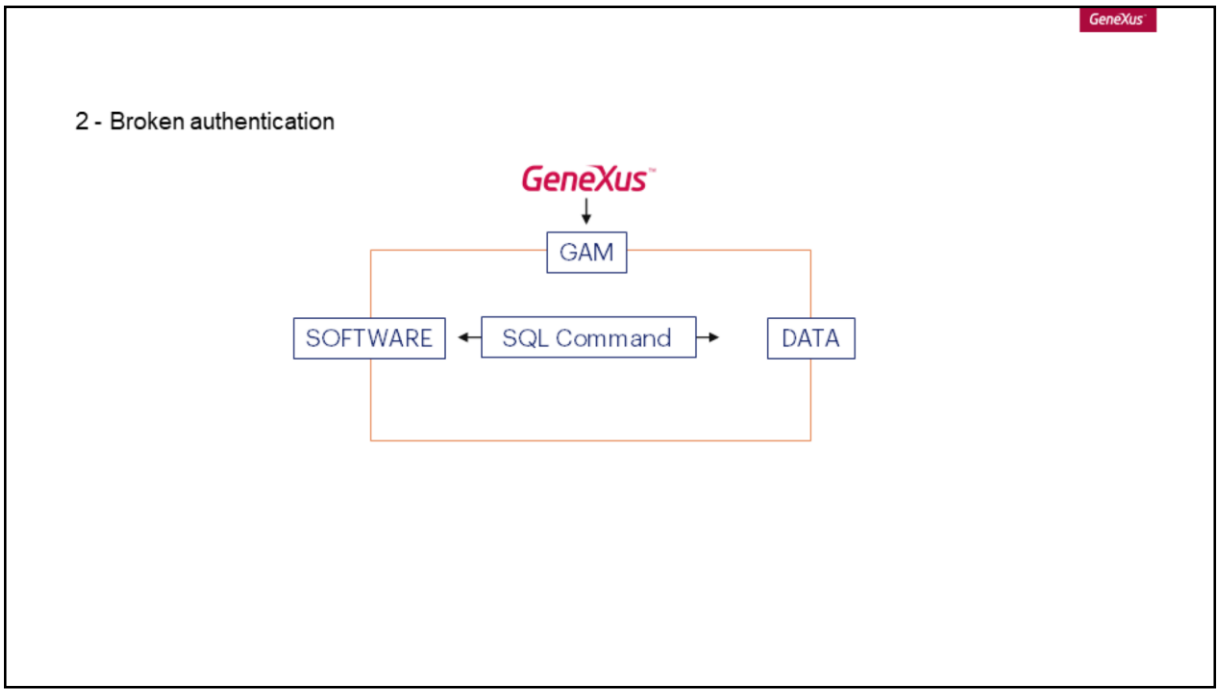
By way of example, with the Top 10 2017 of the Open Web Application Security Project, it is possible to enumerate certain mitigations that GeneXus carries out automatically, and cases where the developer may introduce flaws despite the measures applied by GeneXus.



The first item in the Top 10 2017 are injections.

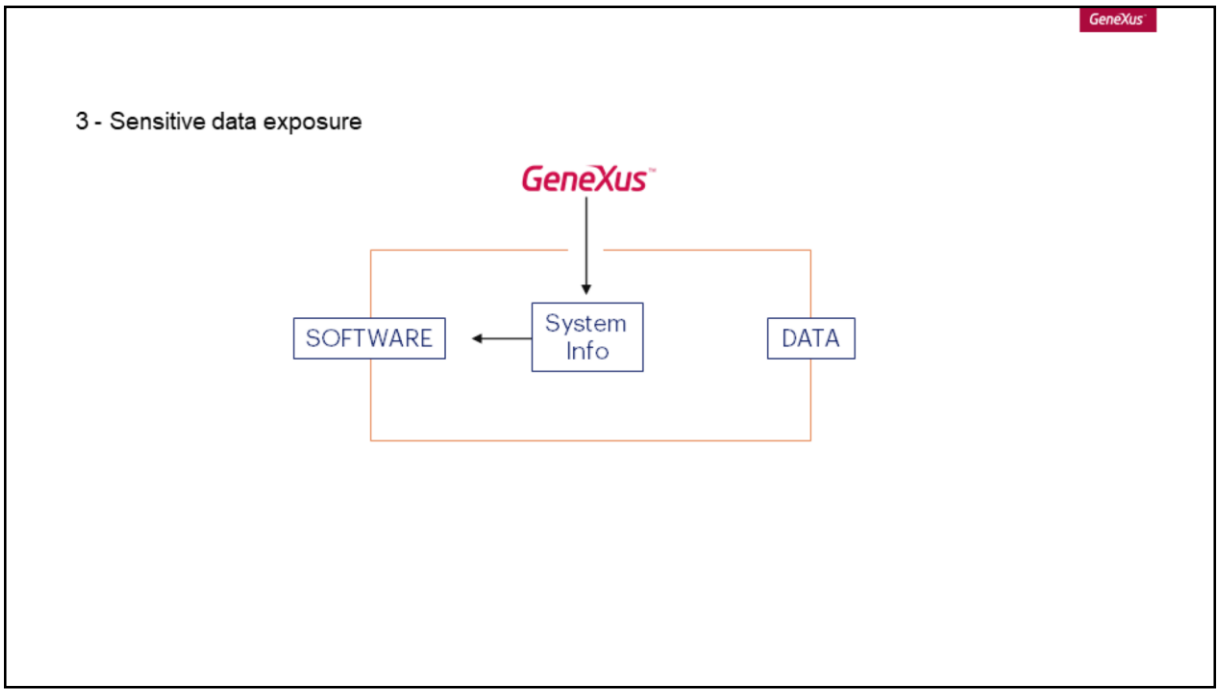
SQL injections are the most renowned, among others. These vulnerabilities take place when information that originates in a user is sent to the database without validation.

GeneXus exposes the SQL Command, which provides the app with increased versatility so that the developer may use the database as a resource. In such case then, the developer is the one who must controls entries of the command in order to prevent an SQL injection.



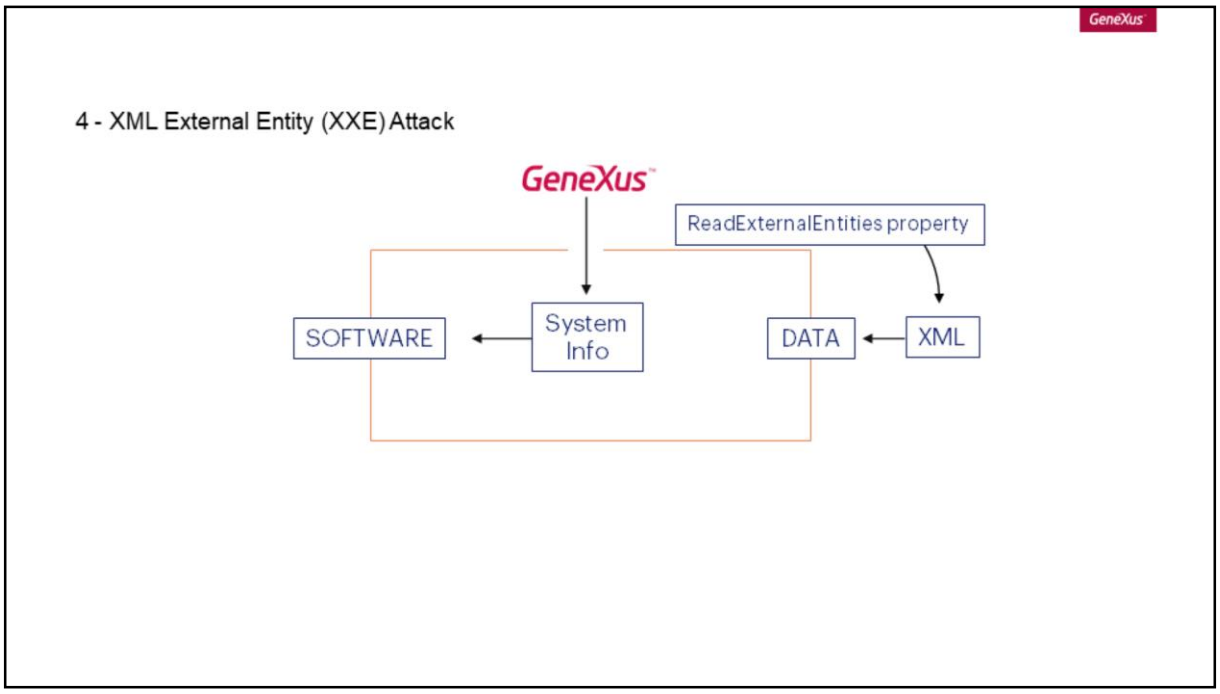
The second item is Broken authentication. GeneXus offers an Authentication/Authorization model called GAM (*Genexus Access Manager*) that allows the automation of such tasks. The proper setup of the tool depends on the developer and on each specific scenario.

It is also possible to use a proprietary module, where the developer is then responsible for managing passwords, sessions, and so on.



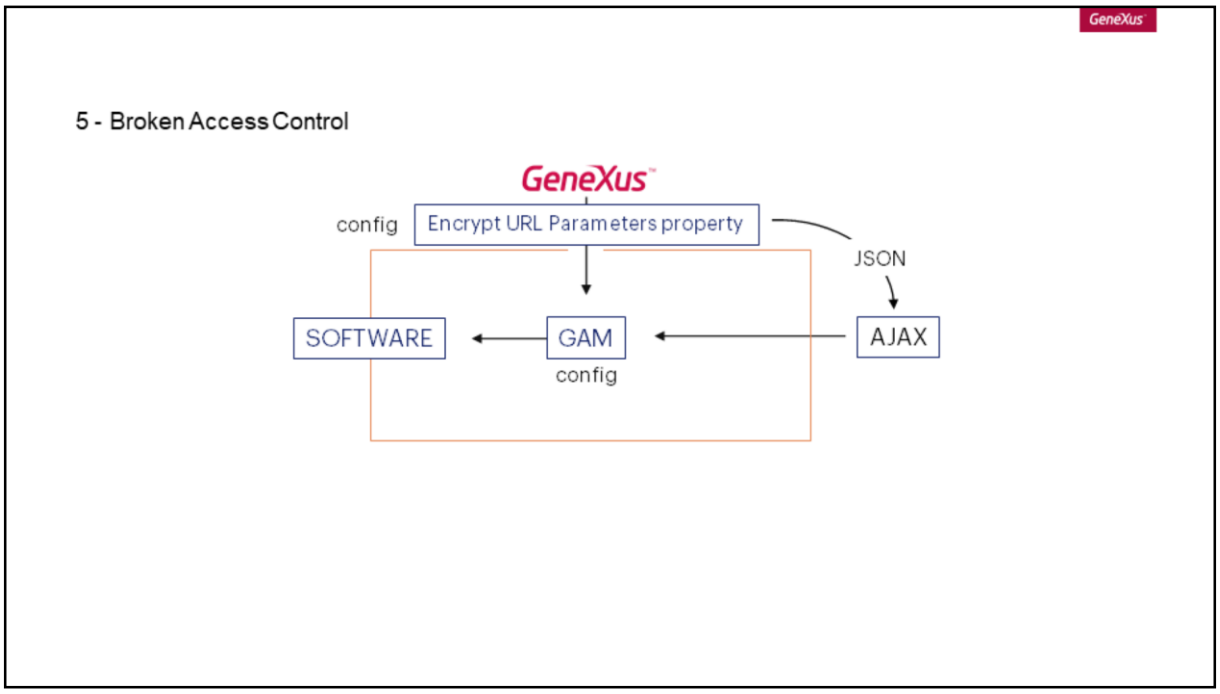
The third item in the Top 10 2017 corresponds to Sensitive Data Exposure. The analysis of requirements includes the determination of the system's sensitive information.

Therefore, GeneXus does not execute any specific action, but it rather provides the functions necessary to protect sensitive information such as ciphering, hashing, and password management and certificates.



The fourth item is an Attack by XML

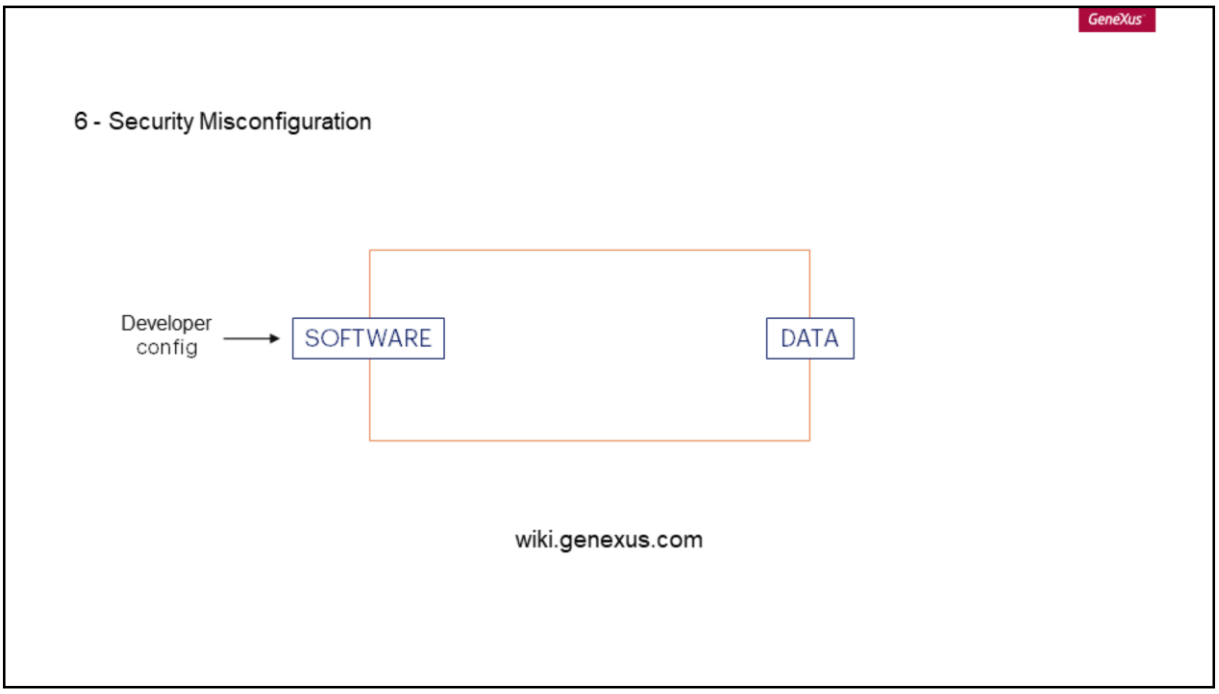
External Entities. In this case, and by default, GeneXus does not process any XML External Entity. In order to enable the processing of External Entities, the ReadExternalEntities property must be set up. In such case, it is also the developer's responsibility to control the XML when the system's border is crossed.



The fifth item corresponds to Broken Access Control, which occurs in cases of non-authorized accesses to the system, alternating URLS, AJAX calls, etc.

To such end, GeneXus includes a property that enables the encryption of URL parameter, where all parameters received by an object are ciphered.

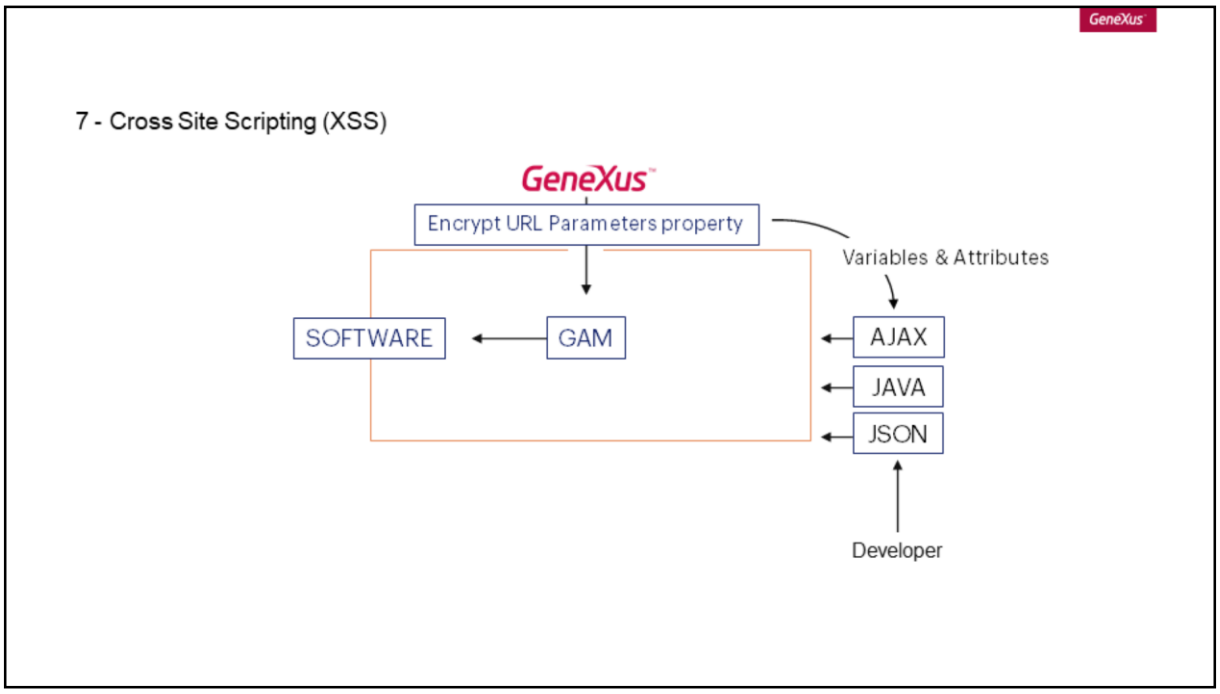
It also controls access with GAM, it executes the codification for JSON in AJAX calls, by default, and it executes validation checks, not only on the client side but also on the server side. In this case, it is the developer's responsibility to add controls to the events, to set up GAM adequately, and to also configure this property that enables the encryption of the URL parameters.



The sixth item in the Top 10 corresponds to Security Misconfiguration and mostly refers to erroneous security configurations in the production applications server. For this case, GeneXus has some properties available that, even when disabling them is not recommendable, a developer may adjust in order to fulfill the provider's needs.

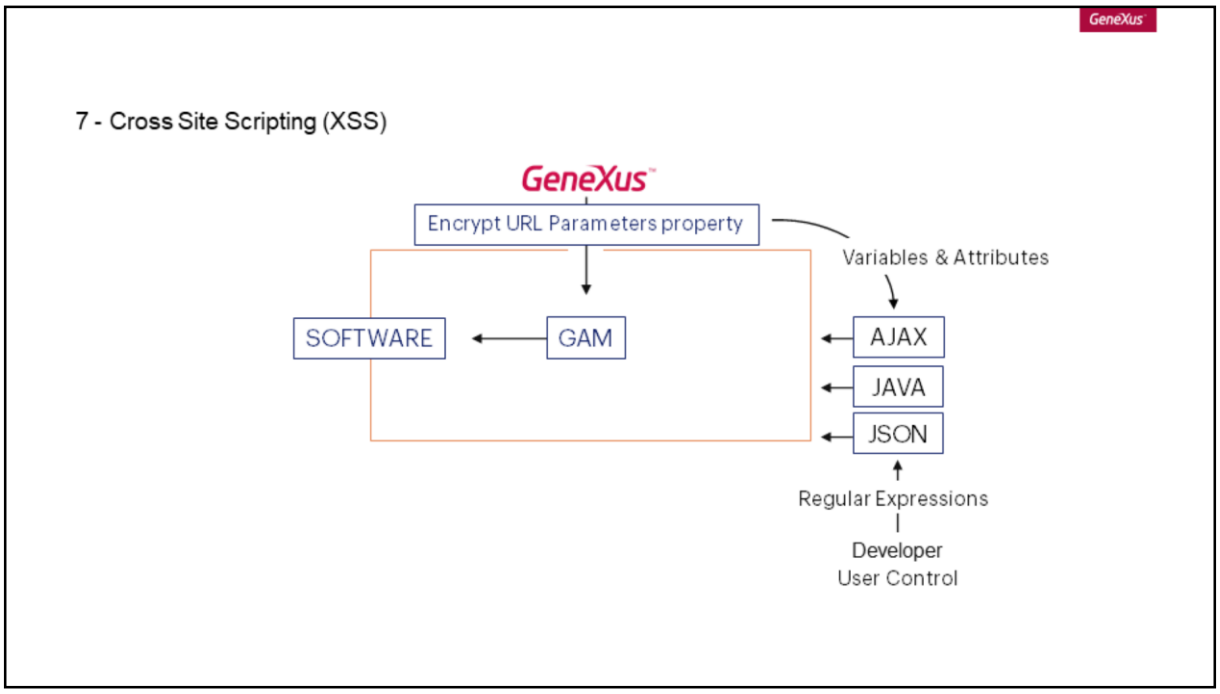
The GeneXus Wiki includes some advice and guidelines for secure server configuration.

However, the degree of security in production will depend, to a greater extent, on the individual in charge of infrastructure who sets up the application's server.

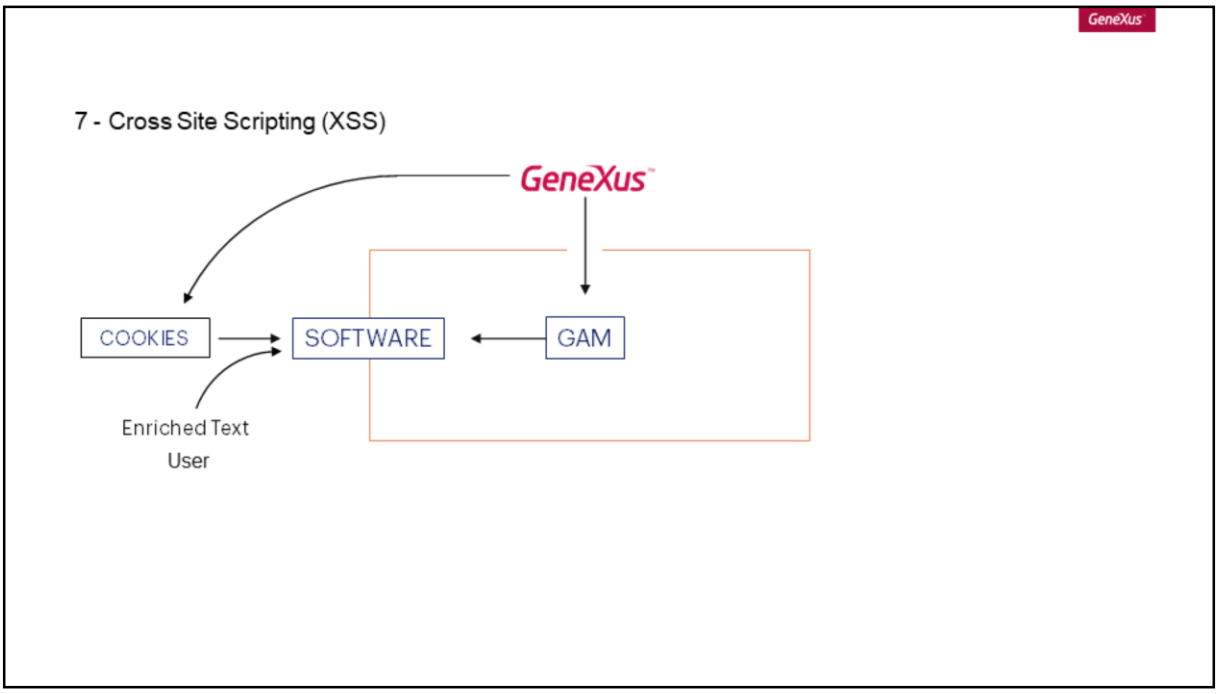


The seventh item in the Top 10 2017 refers to the Sequence of Cross-Site Commands. For this kind of vulnerability, GeneXus automatically validates the length and type of variables and attributes, and it executes the data codification that the system receives according to its context (be it HTML JavaScript, JSON, or other –also automatically).

For cases where the developer introduces his own Javascript or HTML code, the responsibility to validate and control it becomes mandatory.

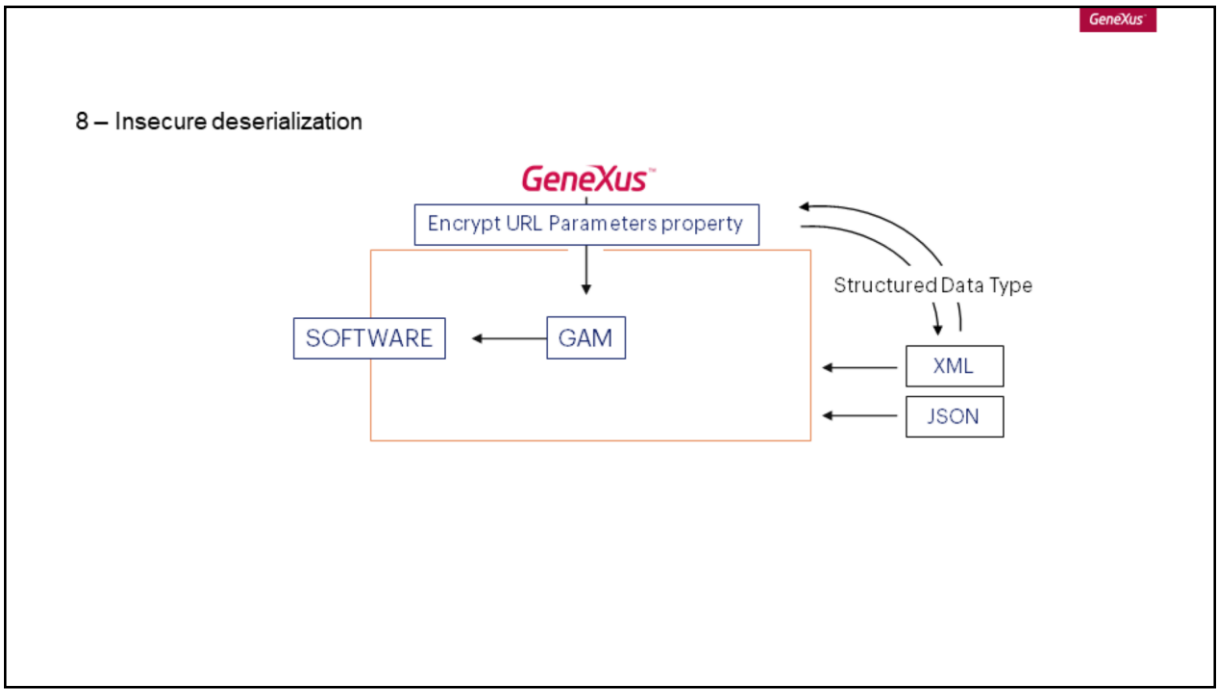


Likewise, GeneXus offers developers the possibility to define Regular Expressions to validate the entries of users or to use a User Control of their own or of third parties (that is, different from GeneXus), to the extent that the developer is totally confident that those entries are absolutely secure and comply with the previously defined validation controls.



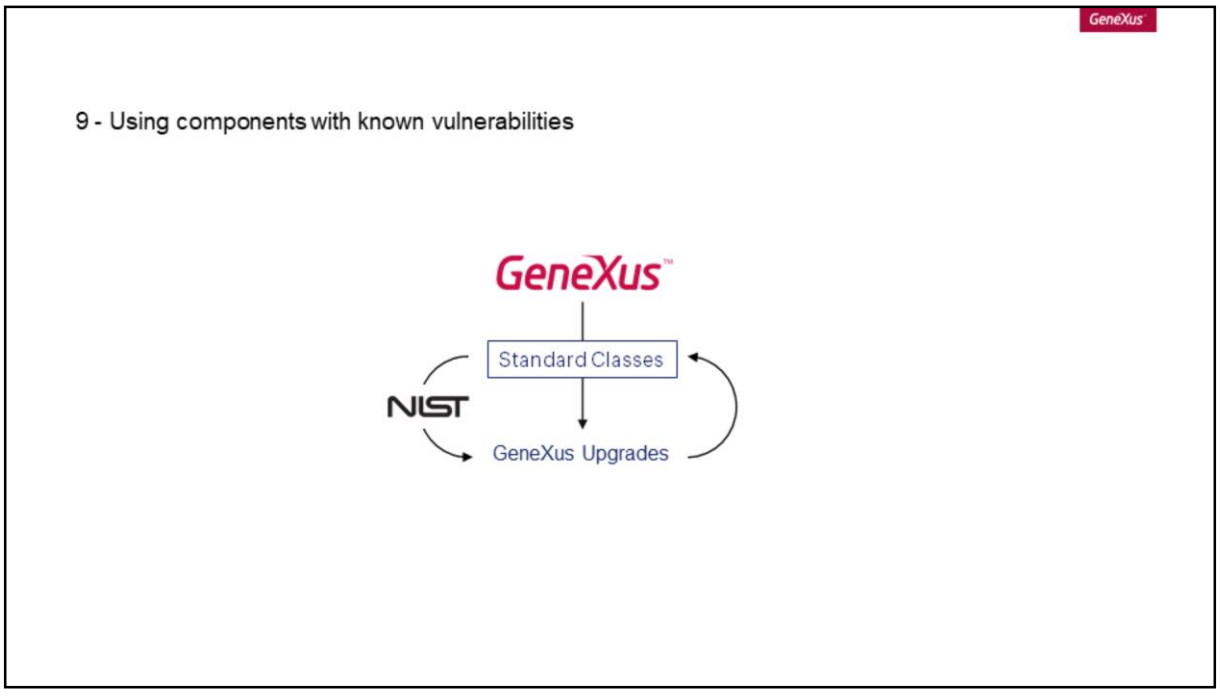
The adequate configuration of cookies must also be borne in mind, and the disabling of checks generated automatically by GeneXus must be avoided (if it is not necessary).

In the case of accepting enriched text from user, we must never forget to control, validate and define the format of this information in a secure and consistent manner.



The eighth item refers to Insecure Deserialization, for which GeneXus implements secure serialization mechanisms in cases that involve structured data types towards JSON and/or XML.

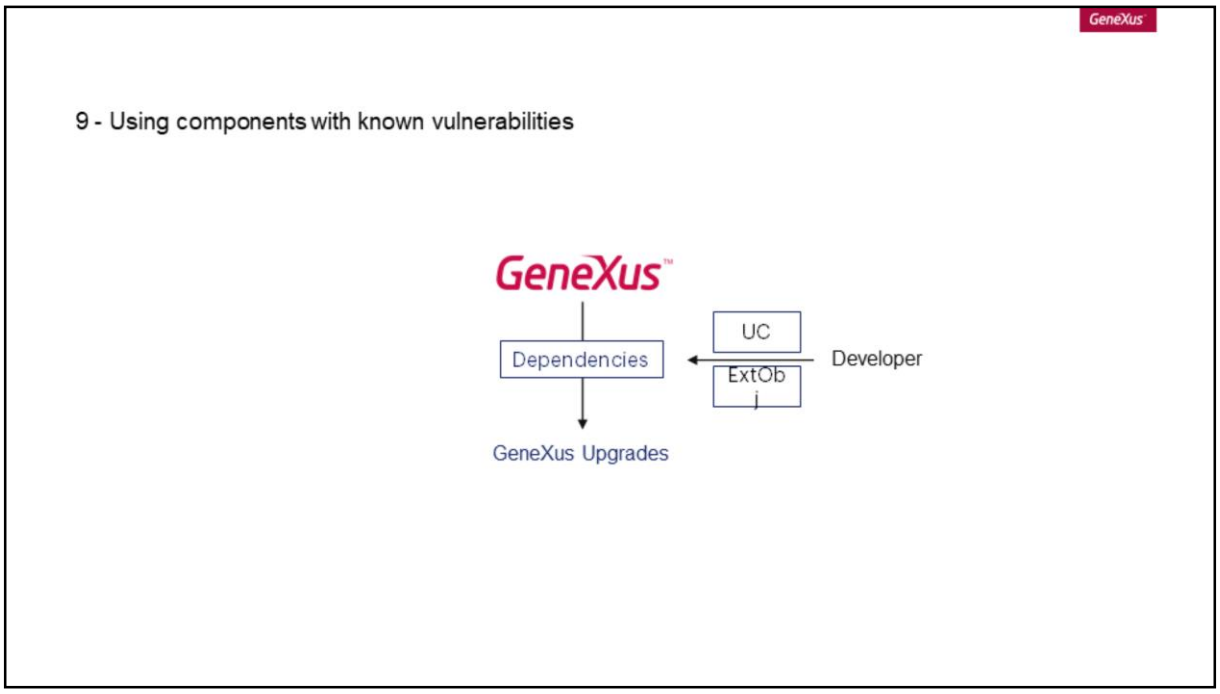
When objects are serialized from XML or JSON using methods or datatypes, entry codification is required.



The ninth item on the list of the Open Web Application Security Project relates to the Use of Components with Known Vulnerabilities.

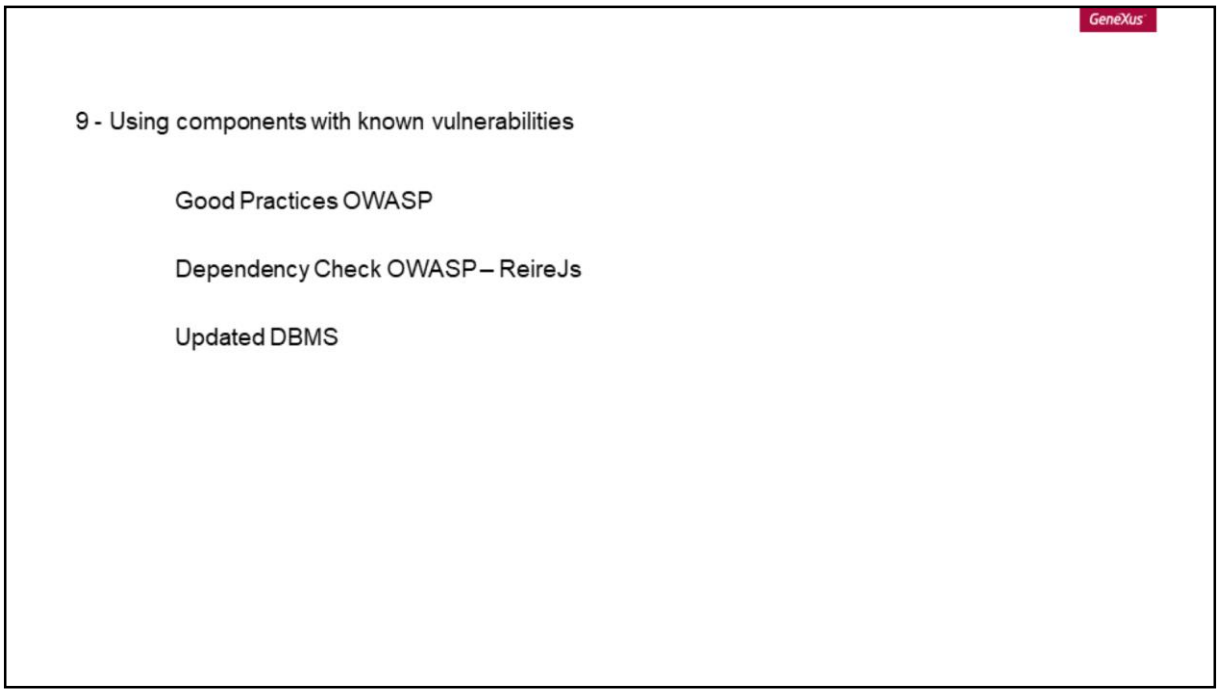
For this purpose, GeneXus has its own standard classes that are permanently reviewed and corrected in subsequent upgrades.

Mas também utiliza dependências de terceiros que são comparadas com a base de dados de vulnerabilidades do Instituto Nacional de Padrões e Tecnologia, sendo atualizadas, se necessário, em cada upgrade do GeneXus.



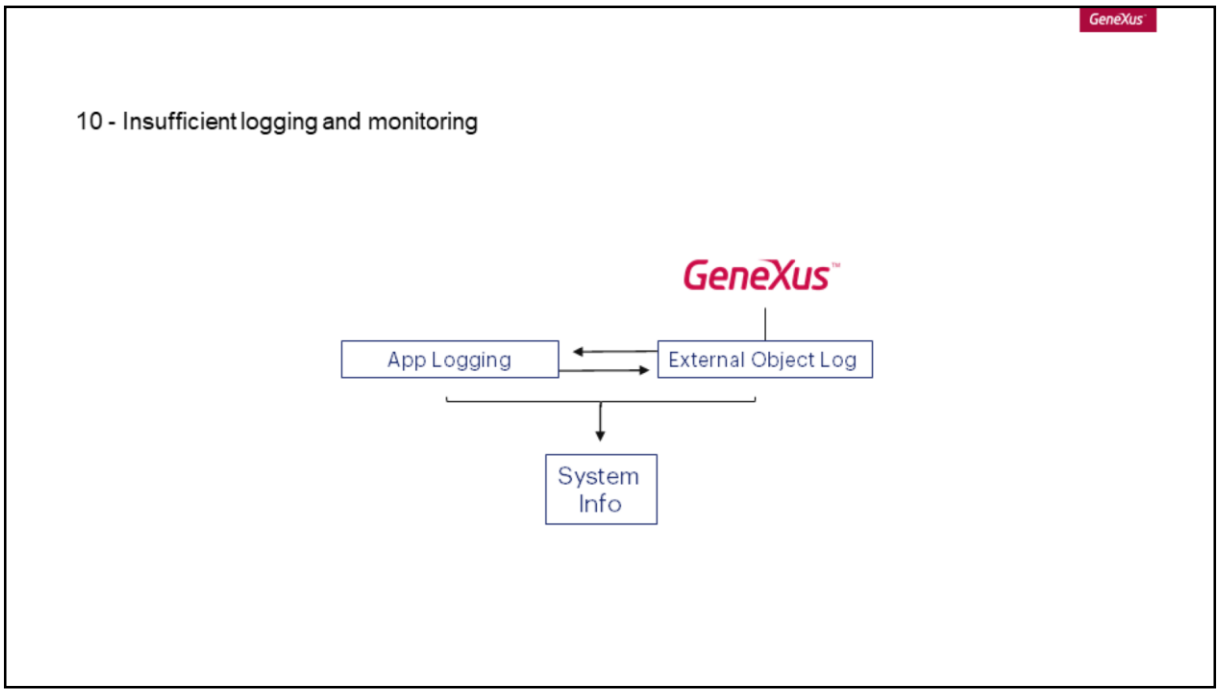
For this purpose, GeneXus has its own standard classes that are permanently reviewed and corrected in subsequent upgrades.

This means that GeneXus takes care of its own dependencies, updating them with the various upgrades. If the developer decides to add a User Control or an External Object, the best practices on secure development included in this Open Web Application Security Project are recommended, as well as reviewing any dependencies included.



To this end, the use of different tools is suggested, in addition to keeping DBMS and their drivers up to date and installing updates and security patches in servers.

.



The last item in the list corresponds to Insufficient Logging and Monitoring.

For all cases, a monitoring process must be defined to detect possible security-related events or incidents.

Although GeneXus offers logging mechanisms at the app level and at the External Object Log level for adding customized information to the application's log, it is necessary to define the relevant information provided by the various devices involved (such as web servers or firewalls). Additionally, the protocols for relieving and categorizing the information gathered must be defined.

This allows for the detection of event that pose a threat to the system and also prevents further incidents.

GeneXus

Security Scanner

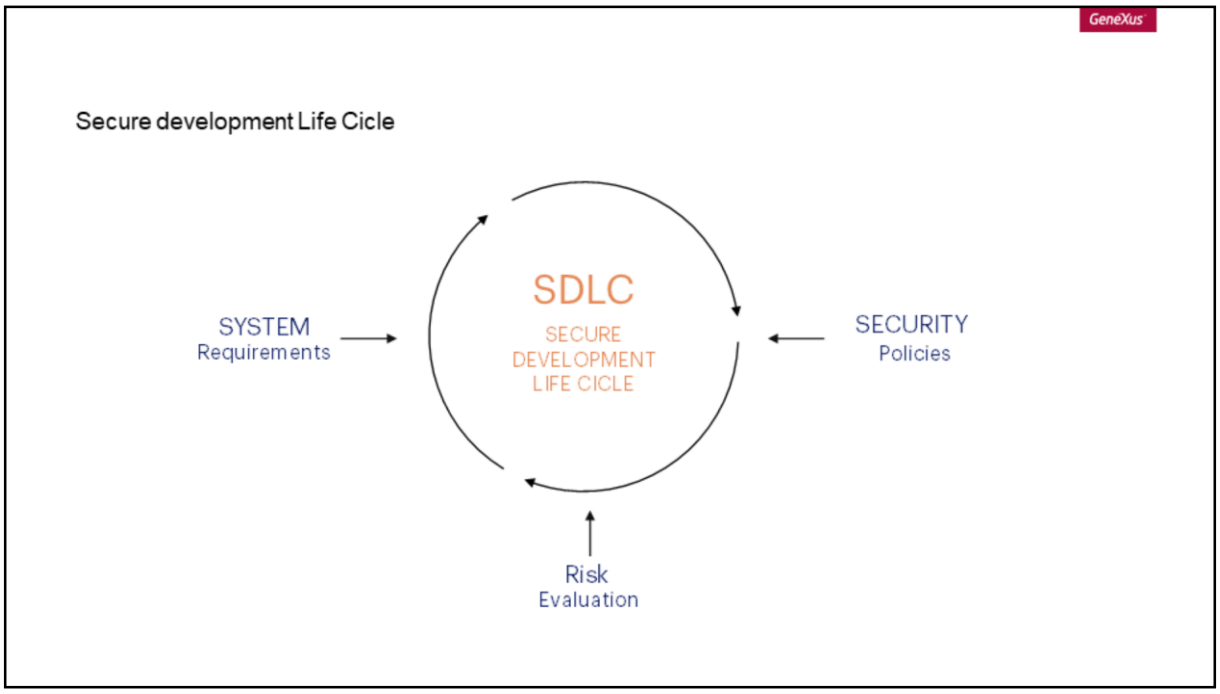
In order to detect known cases where developers might introduce errors, it is possible to use the Security Scanner.

The Security Scanner is an Extension for the GeneXus IDE, available in Marketplace, that executes a static analysis of the knowledge base's objects, searching for traces of security problems already known.

The latest version of the extension is compatible with Genexus 16, and it has been updated to execute a review based on the Top 10 2017 of the Open Web Application Security Project, with documents available in the Wiki.

GeneXus

Secure development Life Cicle



In the development of secure systems, we must implement a **secure life cycle**;

that is, take security into account from the point where the system's requirements are gathered. This is done with a risk assessment based on security policies that should be applied throughout the whole development cycle, in other words: during design, implementation, testing, set-up, and maintenance.

GeneXus™

Videos	training.genexus.com
Documentation	wiki.genexus.com
Certificactions	training.genexus.com/certifications