Тема 8

Сигурност на приложенията

Увод

- Информационната сигурност има много аспекти. Когато се цели(в една организация) информацията да бъде защитена на високо ниво несъмнено трябва се обърне внимание не само на защита от външната среда или мрежата, но и на това каква защита предлагат приложенията, които се използват в ежедневната работа.
- Независимо какви приложения се използват от служителите дали за електронна поща или за счетоводни операции те трябва да се проектират, внедрят и настроят за работа в съответствие с политиките, процедурите и инструкциите за сигурност.
- От друга страна стоят разработчиците на софтуер, които трябва да предлагат приложения, създадени в съответствие на законови изисквания, стандарти и добри практики в областта на сигурността.

Въведение

- Разработката на софтуер е дълъг и сложен процес (Software Development Life Cycle) накратко етапите са:
- Проучване (на предметната област, съществуващ софтуер ...)
- Спецификация на изискванията (Анализ на изискванията)
- Софтуерен дизайн (Софтуерна архитектура)
- Разработка и интеграция на модулите
- Тестване (Валидация)
- Внедряване (Инсталация)
- Поддръжка.

Secure Development Lifecycle Software Security Assurance

- Жизненият цикъл на разработка на сигурни приложения е предмет на внимание от всички водещи разработчици и други организации.
- Microsoft, като един от водещите разработчици работи в тази посока и предлага практики и помощни инструменти за улесняване на разработчиците. (https://www.microsoft.com/en-us/sdl/)
- Oracle е друг водещ разработчик, който предлага методология и инструменти за разработчици (http://www.oracle.com/us/support/assurance/development/secure-coding-standards/index.html).
- Open Web Application Security Project (OWASP) –международна организация, разработва литература, документи, инструменти, форуми и др. OWASP Software Assurance Maturity Model, OWASP .NET, OWASP Mobile Security, OWASP Java https://owasp.org/

Разработка на сигурни приложения

- През целия процес на разработка трябва да се имплементират мерки за сигурност(контроли за сигурност). Започвайки още от идеята за приложението - неговата функционалност следва да се имат предвид и програмата/системата за сигурност на организацията.
- Процесът преминава и през етапите на анализ на изискванията и архитектурата на приложението до тестването и внедряването.
- При етапа на поддръжка най-често разработчиците предлагат на клиентите си именно актуализации за повишаване на сигурността.
- Прилагането на този цикъл на разработка предоставя множество предимства, като:

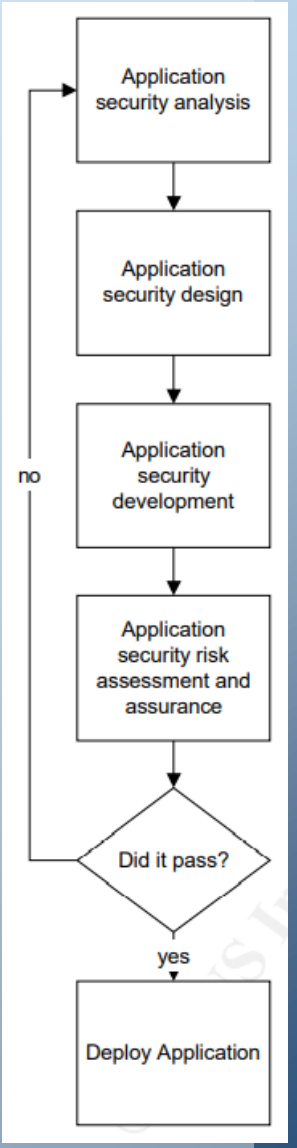
❑ внедряване на механизми за защита за всеки модул;

❑ тестване на тези механизми;

❑ намаляване на разходите и най-вече тези за поддръжка;

❑ високо качество на приложението (добър имидж);

❑ удовлетвореност на клиентите.



Добри практики - обучение

- Обикновено, една програма за обучение по сигурността на екипите програмисти включва техническа информираност относно сигурността, обучение за всички и индивидуално за специфични роли (напр. фронт/бак енд , БД разработчици).
- Специфичното обучение се изразява в по-големи подробности за дейностите и технологиите за сигурност на дадено лице/ длъжност.
- Екипите се запознават с политиките за сигурност на клиента, последните разработки, технологии и средства за осигуряване на сигурност на ниво приложение. Обучението може да включва и запознаване с нормативна уредба и добри практики.
- Препоръките са да се прави обучение поне **веднъж годишно**.

Добри практики - разработка

- Валидиране на входа – проверка на дължината на данните(тел. Номер, ФН), валидиране на кодовата таблица (напр. UTF-8), проверка на формата на данните (дата), използване на ограничения от различен характер (недопустими символи - < > " ' % ( ) & + \ \' \„)
- Автентикация и пароли – използване на TLS за клиентска автентикация, реализация на подходящи съобщения за грешка при автентикация, сигурно съхранение и пренос на паролите криптирани с алгоритъм за еднопосочно кодиране- хеширане Secure hash algorithm (SHA) + salt
- „изчистване на данните“ преди да се предадат на сървъра или на други системи - whitelist (allowlist) , blacklist (blocklist) на допустими символи или стрингове на вход
- Други като: най-малките позволения, отказан достъп по подразбиране, изграждане на слоеве и модули, цифрово подписване на код, сертификати, quality assurance, прод.

Добри практики - разработка

- Контрол на достъпа на ниво запис в БД, а не само на роля
- Криптиране на всички лични данни както при съхранение, така и при преноса
- ID които са уникални идентификатори
- Регистри – кой, кога, какво, колко пъти
- Функция за автоматично известяване на администраторите на системите
- Внимателно използване на библиотеки и компоненти от трети страни или остарели такива
- Използване на инструменти за т.нар. Application Security Testing , напр. Snyk
- Ограничаване на достъпа до проекта/приложението на всички етапи само от отговорния екип

Privacy – поверителност

- Както вече нееднократно се дискутира в този курс поверителността на данните е от изключителна важност за потребителя днес.
- Разработчиците на софтуер следва да създават такива приложения, които отговарят на тези изисквания за поверителност.
- В тази връзка всеки потребител на приложение трябва да бъде известяван за събираната и обработвана информация и такава функционалност трябва да бъде имплементирана.
- Всяко приложение трябва да дава възможност на потребителя да управлява: събирането, обработката и разпространението на информация за самия него.
- Не трябва да се забравя, че всяко приложение трябва да отговаря и на законовите разпоредби, стандартите, фирмените политики и т.н.

Видове данни/информация – по отношение на поверителността (според Microsoft)

- Анонимни – не се отнасят до конкретен човек и не са уникални. Такива могат да бъдат: имена на човек ??? (вижте ревюта на стоки в интернет магазин), статистически данни, методи на плащане и др.
- Псевдоанонимни – уникални идентификатори, които могат да бъдат свързани с определен човек, но все още не са, например администратор на система, модератор на рубрика.
- Персонални – информация, чрез която може да се идентифицира отделен човек: адрес, телефон, имейл и др.
- Чувствителни – особено важна: ЕГН, номер на лична карта, номер на кредитна карта и т.н.

Декларация за поверителност

- Част от почти всеки софтуерен продукт
- Съдържа:
  - **Кой** събира информацията
  - **Каква** информация се събира
  - За какви **цели**
  - **Как** точно се събира, съхранява, обработва и разпространява
  - Как се **достъпва и променя** информация от страна на потребителя
  - Възможности за **избор** пред потребителя
  - Възможности за **сигнали** за нередности

Инфраструктура на средата за разработка

В началото на един нов проект се уточнява:

- хранилищата на изходния код и споделените файлове;
- сървърите за компилиране трябва да бъдат конфигурирани за изключителен достъп на членовете на екипа;
- бъг проследяващия софтуер трябва да бъде конфигуриран да открива бъгове в сигурността само в съответствие с политиките на организацията;
- трябва да се закупят лицензи за софтуерните инструменти за сигурност, ако се използват такива-крайно непрепоръчително е използването на пиратски софтуер. Защо?

Изисквания за сигурност

Изисквания за сигурност могат да включват:

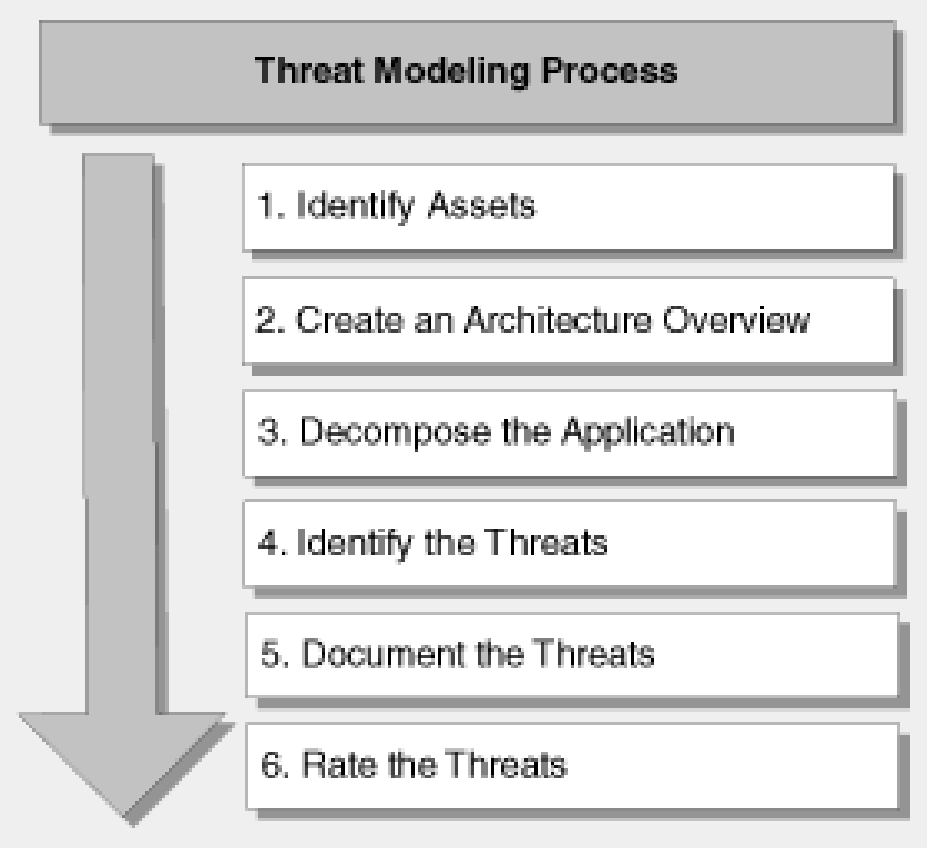
- матрици за контрол на достъпа,
- целите по отношение на сигурността или какво не трябва да се допуска да направи един хакер,
- препратки към политики и стандарти, изискванията за регистриране/лог на събития свързани със проблеми със сигурността,
- изисквания на ниско ниво, като например минимална големина на ключове/пароли или как да се реагира при определени грешки или exceptions.

Моделиране на заплахите

- Представлява систематизирано описание на заплахите и тяхната подредба по това, какъв ефект биха имали върху приложението.
- Позволява да се прилага структуриран подход към сигурността и да се справи с най-честите заплахи, които имат най-голям потенциал за въздействие върху приложението на първо място.
- На второ място осигурява взимането на контра мерки още в процес на проектиране на приложението.
- Могат да се използват софтуерни инструменти, например OWASP Threat Dragon, ThreatModeler, Microsoft Threat Modeling Tool и SecuriCAD.

Пример (моделиране на заплахите)

Майкрософт в добри практики (https://www.microsoft.com/en-us/SDL/process/design.aspx)най-общо описва процеса така:



Сигурно кодиране(програмиране)

- Представлява разработка на приложения по начин, който предпазва от „дупки“ в сигурността.
- Включва използване на библиотеки и функции с доказан произход, прилагане на препоръки и добри практики от специалисти, използване на технологии за сигурност(например криптиране) и други.
- За намиране на пропуски се прилагат например две техники - или се използват инструменти за анализ (Visual Studio code analysis tools или Secure Coding Validation Suite на CERT) или кода на приложението се преглежда от друг специалист.

Тестване

- Тестването по отношение на сигурността е процес, имащ за цел да разкрие недостатъци в механизмите за сигурност на приложението, като защита на данните и работеща функционалност, както е предвидено.
- Благодарение на логическите ограничения на тестване на защитата, това че приложението минава тестването на защитата не е индикация, че не съществуват никакви пролуки или че системата адекватно отговаря на изискванията за сигурност.
- Препоръчително е да бъде извършвано от независими тестери, но първоначално се извършва и от специалистите вътре в компанията разработчик.
- Могат да се прилагат софтуерни инструменти за тестване и тестване от хора (някои компании дори наемат т.нар. етични хакери).

Документация

- Документацията е неизменна част от всяко приложение, в нея са описани начините за инсталиране, настройки и работа с него.
- Тъй като приложението ще бъде използвано от някой друг от екипа за разработка, то потребителя трябва ясно да разбере какво по безопасността на приложението се прилага при инсталирането, какви настройки могат да влияят на сигурността, както и как потребителя да реагира при възникване на всяка грешка.
- Потребителя също така трябва да знае, ако настоящата версия поправя идентифицирани заплахи за сигурността в предишните версии.

Уеб приложения

- С навлизането на Интернет технологиите в бита, образованието, бизнеса, забавленията и др. много от използваните от нас приложения вече са уеб достъпни.
- Например днес е възможно да обработваме текстови документи или електронни таблици без да се налага да инсталираме офис пакети.
- Ако за едно десктоп приложение защитата може да се осигури с по-малко усилия, то за уеб приложенията разработчиците трябва да положат доста повече усилия, да осигурят контролиран достъп и отвън, да защитят информацията, когато тя пътува по мрежата, да осигурят непрекъсната работа на уеб сървъра и много други.
- Разработчиците трябва да обърнат внимание на следните уязвимости: SQL инжектиране, Скриптове и форми, Сесии и бисквитки и традиционни атаки.
- Препоръка е поне използване на инструменти Web application firewalls (WAF), които предпазват от cross-site forgery, cross-site-scripting (XSS), file inclusion, SQL injection и др.

SQL инжектиране

- SQL инжекциите са техника за атакуване на приложения, използващи бази данни, чрез вмъкване на злонамерени SQL команди в заявките, генерирани от приложението.
- Данни, подавани от потребителя винаги трябва да бъдат валидирани от приложението преди да бъдат използвани в SQL конструкции.
- Основната стъпка при защита от SQL инжекции е валидацията на входните данни.
- Валидацията представлява проверка дали входните данни, въвеждани от потребителя, отговарят на очакваните данни от приложението.
- Пример за валидация е проверка дали потребителското име съдържа само символи от A до Z и цифри от 0 до 9, ако само тези символи са заложени като валидни в приложението при въвеждане на потребителско име
- Други мерки са проверка дали въведеното за име не е ключова дума от SQL езика например UNION(Union-based SQL Injection), INSERT или DELETE, филтриране на входните низове за апострофи (‘) и ограничаване на дължината на входните низове.
- Пример: ако се допусне да се въведе " or ""=“ за име и парола ще се върнат всички записи от таблицата с потребителите.
- Пример: http://www.webstore.com/items/items.asp?itemid=123 or 1=1
- Други начини се базират на бисквитките, HTTP headers, Second-order SQL injection

Форми

- Формите в едно приложение са начин на потребителите да въвеждат данни и начин за управление на сесиите и предаване на важни данни по време на сесията, като идентификаторите на потребителя и сесията.
- Атакуващите могат да подменят данните предавани, чрез формите и да измамят приложението за истинския потребител.
- Методи за използване при форми:

➢ Използване на SSL

➢ CAPTCHA Authentication

➢ Деактивирането на скриптове от страна на клиента за предпазване от Cross site scripting (XSS);

➢ Да се избягва предаване на параметри в URL адресите;

➢ Предаване на параметри чрез скрити полета, но не за чувствителни данни.

Клиентски скриптове

- Проблемите със скриптовете при клиента най-често са от вид Javascript injection - cross-site scripting (XSS) или Cross-Site Request Forgery (CSRF)
- Т.нар. JavaScript security holes езика е в първите по уязвимости
- Много от разработчиците използват скриптове (JavaScript) за да валидират въведените от потребителите данни.
- Един злонамерен потребител може лесно да деактивира скриптовете на своя браузър и така да избегне валидацията.
- Разработчиците трябва да отчитат това и да реализират валидацията от страна на сървъра или да реализират така функционалността, че приложението да работи различно, ако скриптовете са деактивирани.
- Трябва да се има предвид обаче, че валидацията от страна на сървъра ще изисква и повече ресурс на самия сървър.

Предаване на параметри в URL адресите

- Чрез използване на форми разработчиците осигуряват предаване на данни от клиента до сървъра по два начина:
- get команда – предава данните в URL
- post команда –предаването е чрез HTTP message body
- ако разработчика използва get за предаване, то става лесно за атака, защото се предава некодирана, освен това обема е ограничен (около 2000 символа)
- използването на post команда е по-сигурно, но пак е възможна атака като се промени целия HTTP пакет

GET vs. POST - https://www.diffen.com/difference/GET-vs-POST-HTTP-Requests

History	Parameters remain in browser history because they are part of the URL	Parameters are not saved in browser history.
Bookmarked	Can be bookmarked.	Can not be bookmarked.
Parameters	can send but the parameter data is limited to what we can stuff into the request line (URL). Safest to use less than 2K of parameters, some servers handle up to 64K	Can send parameters, including uploading files, to the server.
Hacked	Easier to hack for script kiddies	More difficult to hack
Restrictions on form data type	Yes, only ASCII characters allowed.	No restrictions. Binary data is also allowed.
Security	GET is less secure compared to POST because data sent is part of the URL. So it's saved in browser history and server logs in plaintext.	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs.
Restrictions on form data length	Yes, since form data is in the URL and URL length is restricted. A safe URL length limit is often 2048 characters but varies by browser and web server.	No restrictions

Решения при предаване на данни

- Разработчикът може да попречи на нападателите от модифициране на данните, които е трябвало да бъдат скрити, чрез управление на информацията за сесията, използвайки GUID-та, или чрез криптиране на информацията.
- Управление на сесиите. Повечето скриптови технологии от страна на сървъра позволяват на разработчика да съхранява информация за сесията на потребителя и това е най-сигурният начин за запазване на информация, тъй като всички данни се съхраняват локално на уеб сървъра.
- Използване на GUID-та. Могат да се използват за идентификатори на потребителите за ключове на таблици от базите от данни и др. Не могат да се налучкат от хакерите.

Пример http://somesite/changeuserinfo.aspx?user=134

тук идентификатора на потребителя е обикновено цяло число, лесно се разбира начина на даване на идентификаторите, а именно последователни цели числа

и

http://somesite/changeuserinfo.aspx?user=3F2504E0-4F89-41D3-9A0C-0305E82C3301

Сесии и бисквитки

- Уеб сесии се реализират по различен начин при всяка скриптова технология от страна на сървъра, но най-общо те започват, когато потребителят влезе в сайта, и продължават до края, когато потребителят затваря браузър или времето на сесията изтече.
- Сесиите се използват за проследяване на дейностите на потребителите, например, когато се добавят стоки в количката - сайтът съхранява (на сървъра) запис на елементите с помощта на идентификатора на сесията.
- Сесиите могат да се използват съвместно с т.нар. бисквитки (данни, изпратени от уеб сайта, съхранявани от потребителския браузър). Всеки път, когато потребителят посети уеб сайта, който е изпратил бисквитката, браузърът ще изпрати бисквитката обратно към уеб сървъра.
- Сесията използва бисквитките, за да идентифицира потребителя и да го свърже с идентификатора на активната сесия.
- Заплахите идват от възможността бисквитките да се променят, а от там да се компрометира и активната сесия, ако се знае нейния идентификатор. Едно лесно решение е съдържанието на бисквитките да се криптира.
- Друга заплаха е свързана със сесиите и е т.нар. Session hijacking

Други атаки

- Уязвими скриптове и код. Много от разработчиците в желанието си да си спестят време и усилия използват за своите приложения скриптове/кодове, които взимат наготово от непроверени източници(обикновено Интернет). Голяма част от такива кодове крият в себе си и зловреден код.
- Брут-форс логин атаки. Друга често срещана атака е опит многократно да се налучка името и паролата на даден потребител, като се използват т.нар. речници. Противодействието може да бъде ограничение за брой неуспешни логини или по брой опити от IP адрес.
- Препълване на буфера. Нападателя може да опита да изпрати към сървъра много голям по обем данни, които съдържат зловреден код с цел да се изпълни на сървъра и нападателя да получи пълен контрол над сървъра.

Десктоп приложения

- Може да се считат за по-сигурни, но това не винаги е така.
- По-сигурни могат да бъдат, защото се използват само след като се инсталират на даден компютър.
- Ако неоторизиран потребител има достъп до този компютър също може да представлява заплаха.
- Много от атаките са аналогични с тези при уеб приложенията.
- Други са специфични, например декомпилиране и промяна на кода.

Сигурност на клиентските приложения

- Сигурността на приложенията се контролира главно от разработчика на приложението. Администраторът, който поддържа приложенията може да подобри сигурността за някои приложения, но ако приложението не е защитено от разработчика, той не може да направи много само с малко настройки.
- Писане на сигурно приложение е трудно, защото всеки аспект на приложението, като GUI, мрежова свързаност, OS взаимодействие, и управление на данни, изисква обширни познания за сигурност. Не всеки програмист обаче притежава тези познания.
- Много от организациите отчитайки тези особености изискват от своите администратори да приложат всички възможни мерки за повишаване на сигурността.

Администраторски мерки

- „привилегии“ при изпълнение – най-малки
- администрация – подходящи настройки
- актуализации на приложенията – възможно най-често
- интеграция с ОС - механизми за сигурност вградени в ОС
- рекламен и шпионски софтуер – инсталиране на софтуер за защита
- мрежова свързаност – настройки, защитни стени или ограничена – никаква свързаност

Peer-to-Peer приложения

- P2P представлява разпределена мрежова архитектура, съставена от участници, които отдават част от своите ресурси (като процесорно време, дисково пространство, файлове) директно на разположение на другите мрежови участници без необходимост от централна управляваща мрежова инстанция (сървъри).
- Участниците (пиъри) са едновременно доставчици и потребители на ресурси.
- Тези приложения комуникират, като консумират значителна част от мрежовата пропускливост и така затрудняват работата на останалите мрежови приложения, което пък може да застраши сигурността.
- Друга опасност се крие във файловете, които си обменят пиърите. Те могат да съдържат вируси, нелицензирано съдържание и др.

Заключение

- По отношение на сигурността в най-общ план приложенията могат да се разделят на „добри“ и „лоши“.
- Във всяка организация, хората, които отговарят за сигурността, трябва да ограничат използването на приложения, които могат да застрашат сигурността, чрез прилагане на съответните мерки.
- На служителите трябва да се забрани да инсталират и пренастройват приложения без нужната квалификация.
- Всички служители следва да се запознаят с политиките на организацията касаещи използваните приложения.

Практически упражнения

- В рамките на един учебен час потърсете в Интернет пространството информация за разглежданите в темата понятия, дефиниции и аспекти. Търсенето може да направите и на чужди езици, които владеете.
- Анализирайте намерената информация и я сравнете с поднесената тук.
- Проучете работните рамки и езици за програмиране с които сте запознати какви механизми за сигурност предлагат.