

**Технически Университет – София**

**Факултет Приложна Математика и Информатика**

**Катедра Информатика**

**КУРСОВ ПРОЕКТ**

**Тема:** Анализ на уязвимости и атаки касаещи груповия проект. Реализация на превантивни мерки

**Имена на студентите:** Станислав Бисеров Стоянов

**Факултетен номер:** 471218066

**Група:** 76

**Съдържание:**

1. Увод
2. Пароли (Brute force attack, Dictionary attack)
3. Подслушване на трафика (Packet sniffing, Man in the middle)
4. Спиране на услугите (HTTP floods, Slowloris)
5. Изпълнение на зловреден код (Cross site scripting)
6. Заразяване на сървъра с вируси (Ransomeware)
7. База данни (SQL injection)
8. Кражба на идентичност (Cross site request forgery)
9. Източници

*1. Увод*

Уеб сигурността е известна още като “киберсигурност” и по същество означава: “защита на уебсайт или уеб приложение чрез откриване на уязвимости, предотвратяването им и превенция срещу кибер заплахите”. Уеб сайтовете и уеб приложенията са също толкова податливи на пробиви в сигурността, колкото нашите физически домове, магазини и държавни институции. За съжаление, киберпрестъпността се случва всеки ден и са необходими големи усилия, средства и ресурси за уеб защита. Ето защо се развива доста бързо направлението уеб защита – система от мерки за защита и протоколи, които могат да предпазят вашия уебсайт или уеб приложение от хакване или неоторизиран достъп. Всичко, което се използва в интернет, трябва да има някаква форма на уеб защита. В този курсов проект представям едни от най-сериозните уязвимости и атаки както и тяхната реализация в реално работещо уеб приложение – **CookingHub**[[1]](#footnote-1).

*2. Пароли (Brute force attack, Dictionary attack)*

**2.1. Brute force attack**

* Определение

Брут форс атаката (още метод на грубата сила) е метод за получаване на почти всякакъв вид тайна дигитална информация. Тя се базира на метод, при който атакуващият тества множество от предварително дифинирани стойности, докато намери правилните. Най-често за този процес се използва автоматизиран софтуер, който по зададени правила генерира тези стойности и/или използва и набор от книжовни думи от речниците, за да тества дали паролата ви не съдържа някоя съществуваща дума.

Този тип атака се използва най-вече за достъпване на доста по-сериозна информация, а именно за разбиване на паролите на административните профили. Като например за достъп до административния панел на вашия сайт, било то WordPress, Joomla, OpenCMS или друг вид CMS (система за управление на съдържанието).

* Как да се защититим?

1. Използвайте необичайни данни за достъп, а именно потребителско име различно от “admin”, “user”, “administrator” и други, както и силна парола. Силната парола включва голям брой букви, цифри и специални символи, които нямат определен логически смисъл. Трябва да се избягват пароли от рода на “123456”, “123456123” и др.

2. Използвайте сървърна защита, която включва:

* Необичайни адреси за достъп до системата
* Пример за обичаен адрес: <http://yoursite.com/admin>
* Пример за необичаен адрес: <http://yoursite.com/dostap>
* Защита на файловете, отговарящи за обработката на входните данни, чрез ограничаване на IP адресите, които имат достъп до тези файлове
* Реализация на защита срещу атаката в проекта
* Добавена е валидация на паролата въведена от потребителя в регистрационната форма, която включва:
* необходимо е наличието на цифра в паролата
* необходимо е наличието на малка буква
* необходимо е наличието на голяма буква
* необходимо е наличието на специални символи
* минималната дължина на паролата е 8 символа

**2.2. Dictionary attack**

* Определение

Атаката на речника е метод за проникване в защитен с парола компютър или сървър чрез системно въвеждане на всяка дума от речника като парола. Атака в речника може да се използва и в опит да се намери ключът, необходим за дешифриране на криптирано съобщение или документ.

Речниковите атаки работят, защото много потребители и фирми настояват да използват обикновени думи като пароли. Речниковите атаки рядко са успешни срещу системи, които използват фрази с много думи, и неуспешни срещу системи, които използват случайни комбинации от главни и малки букви, смесени с цифри. В тези системи методът за атака с груба сила (при който всяка възможна комбинация от символи и интервали се изпробва до определена максимална дължина) понякога може да бъде ефективен, въпреки че този подход може да отнеме много време и ресурси, за да даде резултати.

* Как да се защитим?

1. Силни пароли: това е една от най-лесните, но най-ефективните превантивни мерки, които човек би могъл да използва. Чрез увеличаване на сложността на вашата парола можете лесно да се борите с речникови атаки. Сложна, уникална и дълга парола с буквено-цифрови символи не се среща в речниците и е трудно да се отгатне.

* Минималната дължина на вашата парола трябва да бъде 8 знака.
* Тя трябва да съдържа както малки, така и главни букви.
* Паролата ви трябва да включва цифри.
* Използването на специални символи е задължително.

2. Редовно променяйте паролите си: редовната промяна на паролите ви също ще ви предпази от речникови атаки. Много организации на ниво предприятие изискват от вас да сменяте редовно паролата на акаунта си, същото трябва да се следва и от домашните потребители. Смяната на паролата на всеки 30 дни може да ви помогне за по-добра защитеност на профила.

3. Блокиране на потребителя при неуспешни опити за вход в системата: Този метод включва деактивиране на акаунта ви след няколко неуспешни опита за влизане. Това създава пауза между всеки опит и по този начин е необходимо повече време и ресурс за хакерите да познаят паролата ви.

* Реализация на защита срещу атаката в проекта
* реализацията на т.1 е представена като мярка и при Brute force атаката
* реализацията на т.2 е имплемнтирана на ниво ASP[[2]](#footnote-2) в проекта, като за целта потребителят се уведомява на всеки 30 дни за смяна на акаунтската парола
* реализация на т.3 чрез имплементиране на lockout (заключваща) функционалност – максималният брой на неуспешни опити за вход в системата е 10, като след блокиране на акаунта, той се отключва след 10 минути

*3. Подслушване на трафика (Packet sniffing, Man in the middle)*

**3.1. Packet sniffing**

* Определение

“Packet sniffing” е практика за събиране и регистриране на някои или всички пакети, които преминават през компютърната мрежа, независимо от това как е адресиран пакетът. По този начин всеки пакет или определена подгрупа от пакети може да бъде събрана за по-нататъшен анализ. Вие като мрежови администратори можете да използвате събраните данни за най-различни цели като наблюдение на честотната лента (bandwidth) и трафика.

“Packet sniffer”, понякога наричан анализатор на пакети, се състои от две основни части. Първата е мрежов адаптер, който свързва “sniffer” със съществуващата мрежа. Втората пък е софтуер, който осигурява начин за регистриране, преглед или анализ на данните, събрани от устройството.

Има два основни вида подслушвания: Hardware Packet Sniffers и Software Packet Sniffers.

* Как да се защитим?

За да бъде едно уеб приложение защитено от такава подслушваща атака, фирмите и крайните потребители трябва да се пазят от приложения (уеб сайтове), които използват несигурни протоколи, като основно HTTP, File Transfer Protocol (FTP) и Telnet. Вместо това трябва да се предпочитат защитени протоколи като HTTPS, Secure File Transfer Protocol (SFTP) и Secure Shell (SSH). В случай, че е необходимо да се използва някакъв несигурен протокол, цялото предаване на данни трябва да бъде криптирано. Ако е необходимо, VPN (виртуални частни мрежи) може да се използва за осигуряване на защитен достъп на потребителите.

* Реализация на защита срещу атаката в проекта
* Използване на HttpsRedirection Middleware чрез промяна конфигурацията на кода в Startup.cs класа на проекта. Чрез този Middleware всички HTTP заявки се пренасочват към HTTPS протокола.

**3.2. Man in the middle**

* Определение

Man in the middle (посредник) – това е вид кибер атака, чиято цел е незабелязано внедряване в комуникацията между две системи и пренасочване на трафика през трета страна. Така хакерът може да променя, изпраща и получава данни предназначени за друг получател.

MITM атаките представляват голяма заплаха за сигурността, защото дават възможност за прихващане на важен трафик и замяна на съдържанието му в реално време. Чрез този тип атаки нападателят има контрол върху цялата комуникация между страните.

Най-често използваният метод за прилагане на атака от тип MITM е чрез разпространение на зловреден софтуер, който дава достъп до данните на уеб браузърите на целевата система. По този начин става възможно контролирането и променянето на уеб информацията, която се приема и изпраща от заразеното устройство.

Чрез тази информация потребителите лесно могат да бъдат препратени на фалшив дубликат на желания от тях сайт или трафика им да преминава през proxy сървър, създаден от нападателя, преди да се достигне до крайната цел – сайт. Така хакерът би могъл да открадне чувствителна информация като пароли, лични данни, банкови сметки и др. Честа цел за такъв тип атаки са онлайн магазини – или по-точно данните за клиентите им.

* Как да се защитим?

1. Уверете се, че вашите служители не използват обществени мрежи.
2. Използвайте VPN (виртуална частна мрежа), за да осигурите сигурни връзки на вашата организация.
3. Защитете имейлите си, като използвате SSL/TLS. Освен това можете да помислите и за PGP/GPG криптиране.
4. Редовно правете анализ на мрежите и устройствата си. Наблюдавайте активността, за да можете незабавно да забележите необичайни дейности.
5. Не забравяйте да актуализирате браузърите си. Уверете се, че вашата организация винаги използва най-новите версии на защитени браузъри.
6. Отделете своите Wi-Fi мрежи. Уверете се, че гостите не използват вашата вътрешна мрежа.
7. Инсталирайте високотехнологични, способни системи за откриване на прониквания във вашата система.
8. Внедрете двуфакторно удостоверяване (two-factor authentication).

*4. Спиране на услугите (HTTP floods,* *Slowloris)*

**4.1. HTTP floods**

* Определение

HTTP flood е вид обемна разпределена атака за отказ на услуга (DDoS), предназначена да “затрупа” целевия сървър с HTTP заявки. След като сървърът е натоварен с обработването на много заявки и не е в състояние да отговори, възниква отказ на услугата за допълнителни заявки.

HTTP flood е вид DDoS атака от „слой 7“. Слой 7 е приложният слой на модела OSI и се отнася до интернет протоколи като HTTP. HTTP е основен интернет протокол и често се използва за зареждане на уеб страници или за изпращане съдържанието на формуляри (форми). Предотвратяването на атаките на приложния слой е особено сложно, тъй като зловредният трафик е трудно да се разграничи от нормалния трафик.

За да постигнат максимална ефективност, злонамерените участници обикновено използват или създават бот-мрежи (botnets), за да увеличат максимално въздействието на атаката си.

Има две разновидности на HTTP flood атаки:

* HTTP GET атака - при тази форма на атака множество компютри или други устройства са координирани за изпращане на множество заявки на изображения или файлове. Когато таргетираният сървър е затрупан с входящи заявки, на които трябва да отговори, възниква отказ на услугата и той вече не може да получава нови заявки.
* HTTP POST атака - обикновено, когато формуляр е изпратен на уебсайт, сървърът трябва да обработи входящата заявка и да изпрати данните в persistence слой, най-често база данни. Процесът на обработка на данните от формуляра и изпълнението на необходимите заявки на базата данни е относително интензивен в сравнение с количеството процесорна мощност и честотна лента (bandwidth), необходими за изпращане на POST заявката. Тази атака използва несъответствието в относителното потребление на ресурси, като изпраща много POST заявки директно до целевия сървър, докато настъпи отказ на услугата.
* Как да се защитим?

Както бе споменато по-рано, предотвратяването на атаки от седмият OSI слой е сложно и трудоемко. Един от методите е да се провери дали заявката, която се изпраща към сървъра идва от бот, подобно на тест за captcha.

Други начини за спиране на HTTP flood атаки включват използването на защитна стена за уеб приложения (WAF), управление на база данни за IP репутация с цел проследяване и селективно блокиране на злонамерен трафик, както и текущ анализ от инженери. Услугите, които се предлагат от Cloudflare[[3]](#footnote-3) анализират трафика от различни източници и редуцират потенциалните атаки.

**4.2. Slowloris**

* Определение

Slowloris е вид атака за отказ на услугата, която позволява на една машина да свали уеб сървъра на друга машина с минимална честотна лента и странични ефекти върху несвързаните услуги и портове. Slowloris атаката се опитва да поддържа отворени много връзки с частична заявка към целевия уеб сървър и да ги държи отворени възможно за най-дълго време. Периодично се изпращат последващи HTTP хедъри, като добавка, но никога не се изпраща цялата заявка (request). Засегнатите сървъри поддържат тези връзки отворени, запълвайки своя connection pool, като в крайна сметка отказват допълнителни опити за връзка от страна на клиентите (браузърите). Този тип атака е кръстена на бавни лорита, група примати, които са известни със своето бавно движение.

* Как да се защитим?

1. Увеличете максималния брой клиенти, които уеб сървърът може да обработи
2. Ограничете броя на връзките, които може да се пратят от един ИП адрес
3. Поставете ограничение върху минималната скорост на трансфер при свързване
4. Ограничете времето, през което клиентът има право да поддържа връзка.

*5. Изпълнение на зловреден код (Cross site scripting)*

**5.1. Cross site scripting**

* Определение

Cross-Site Scripting (XSS) атаките са вид “инжекция”, при която злонамерени скриптове се изпълняват в иначе доброкачествени и надеждни уебсайтове. XSS атаките възникват, когато нападателят използва уеб приложение, за да изпрати злонамерен код обикновено под формата на скрипт от страна на браузъра до различен краен потребител. Пропуските в системите, които позволяват на тези атаки да бъдат успешни, са доста широко разпространени и се появяват навсякъде, където уеб приложението очаква входни данни от страна на потребителя и тези данни не се валидират и изчистват (sanitize - санитизират) от злонамерения код.

Нападателят може да използва XSS, за да изпрати злонамерен скрипт до нищо неподозиращ потребител. Браузърът на крайния потребител няма начин да разбере, че на скрипта не трябва да се вярва и ще го изпълни. Тъй като браузърът смята, че скриптът е дошъл от надежден източник, зловредният скрипт може да има достъп до всякакви бисквитки, сесиини токъни или друга чувствителна информация, запазена в браузъра и използвана в даденото уеб приложение. Тези скриптове могат дори да пренапишат съдържанието на HTML страницата. XSS атаките обикновено могат да бъдат разделени на две категории, като съществува и трети много по-малко известен тип, наречен DOM базиран XSS:

* съхранявани (stored)
* отразени (reflected)
* Съхранявани XSS атаки

Съхраняваните атаки са тези, при които инжектираният скрипт се съхранява постоянно на целевите сървъри, например в база данни, във форум, дневкник на посетителите (visitor log), поле за писане на коментар. След това жертвата извлича злонамерения скрипт от сървъра, когато поиска дадена информация от него (изпраща request). Съхраняваните XSS атаки се наричат още и постоянни (Persistent) или атаки от тип 1.

* Отразени XSS атаки

Отразените атаки са тези, при които инжектираният скрипт се отразява от уеб сървъра, например в съобщение за грешка, резултат от дадено търсене или друг отговор (response), който включва част или целия вход, изпратен към сървъра като request. Отразените атаки се изпращат на жертвите по друг път, например в имейл съобщение или като препратка към друг уеб сайт. Когато потребителят кликне върху злонамераната връзка, попълни специално създадената злонамерена форма или просто прегледа злонамерения сайт, инжектираният скрипт се изпраща до уязвимия уеб сайт, който обратно отразява атаката към браузъра на потребителя. След това клиентският браузър изпълнява скрипта, защото е дошъл от “достоверен” сървър. Отразените XSS атаки се наричат още и непостоянни (Non-Persistent) или атаки от тип 2.

Примери за XSS атаки:

* XSS атаките могат да се извършват и без използване на таговете <script>…</script>. Останалите тагове вършат абсолютно същото, като <body onload=alert(‘test1’)> или атрибутите като: onmouseover, onerror.

**onmouseover**

<b onmouseover=alert('Wufff!')>click me!</b>

**onerror**

<img src="http://url.to.file.which/not.exist" onerror=alert(document.cookie);>

* Ако трябва да преминем през филтрите на уеб приложенията, може да се опитаме да кодираме следните символи: a = &\#X41 (UTF-8) и да го използваме в таговете за снимка – img:

**<IMG SRC=j&#X41vascript:alert('test2')>**

* Как да се защитим?

1. Никога не поставяйте ненадеждни данни във вашия HTML вход, освен ако не следвате следните изброени стъпки. Недостоверните данни са всички данни, които могат да бъдат контролирани от хакера, входни HTML формуляри, URL низове (query strings), HTTP хедъри, дори данни, получени от базата данни.
2. Преди да поставите някакви ненадеждни данни в структурата на HTML кода, се уверете, че те са кодирани (HTML encoded). HTML кодирането взима символи като “<” и ги променя в безопасна форма като &lt;.
3. Преди да поставите ненадеждни данни в JavaScript кода, поставете тези данни в HTML елемент, чието съдържание се извлича по време на изпълнение на кода. Ако това не е възможно, се уверете, че данните са кодирани в JavaScript. JavaScript кодирането взима символи като “<” и ги кодира в шестнайсетичния им формат, тоест “<” се преобразува в \u003C.
4. Преди да поставите ненадеждни данни в URL, се уверете, че те са кодирани.
5. Чрез използване на HtmlEncoder, JavaScriptEncoder и UrlEncoder

*6. Заразяване на сървъра с вируси (Ransomeware)*

**6.1. Ransomeware**

* Определение

Ransomware е злонамерен софтуер, който шифрова файловете ви или ви спира да използвате компютъра си, докато не платите пари (откуп), за да бъдат отключени. Ако вашият компютър е свързан към мрежа, Ransomware може да се разпространи и на други компютри или устройства за съхранение в мрежата. Някои от начините да се заразите с Ransomware включват:

* Посещение на опасни, подозрителни или фалшиви уеб сайтове.
* Отваряне на прикачени файлове, които не сте очаквали или от хора, които не познавате.
* Отваряте злонамерени или лоши връзки в имейли, Facebook, Twitter и други социални мрежи, мултимедийни съобщения или чрез SMS чатове.

Често можете да разпознаете фалшив имейл или уеб страница, защото имат лош правопис или просто изглеждат необичайно. Внимавайте за странни начини на изписване на имена на компании (като „PayePal” вместо „PayPal”) или необичайни интервали, символи или препинателни знаци (като „iTunesCustomer Service”, вместо „iTunes Customer Service”). Ransomware може да се насочи към всеки компютър, независимо дали това е домашен компютър, компютри в мрежа на предприятие или сървъри, използвани от държавна агенция. Една от най-големите грешки, които допускат собствениците на уеб сайтове, е да си мислят, че такъв тип атака никога няма да им се случи. Ransomware е популярен сред много хакери, защото е злонамерен тип доход, а не просто унищоване на данни за удоволствие. Това е просто бизнес, който е много добре подготвен и ефективен. Всички са изложени на риск от този тип атака, особено ако имате някакъв тип публична платформа. Прекрасен пример е градският транспорт в Сан Франциско. Извършителят на атаката поискал 73 000 долара след блокиране на билетните системи на общинската агенция по транспорт (Minicipal Transportation Agency). Многобройни други атаки продължават да се случват и цените на откупите варират от 570 долара до 17 000 долара.

* Как да се защитим?

Ransomware атаките много трудно се митигират и ограничават. Веднъж завзели едно компютърно устройство, те не могат да бъдат спрени и жертвите трябва да заплатят немалки суми, за да разкодират обратно своите файлове. Тъй като заплащането се осъществява в биткойни (bitcoin – най-използваната виртуална валута на Интернет), става още по-трудно идентифицирането на създателите на тези вируси и тяхното залавяне. Още по-лошото е, че дори да платите има голяма вероятност вашите файлове да не бъдат възстановени. Това е така, защото имейл адресите, които се използват от кибер престъпниците, биват спирани бързо от доставчиците на определените имейл услуги и те не могат да получат уведомителните електронни писма за извършеното плащане.

За да се предпазите от криптиращи вируси и зловредни софтуери, когато сърфирате в Интернет или използвате “Мрежата за споделяне и сваляне на файлове”, се отнасяйте с внимание към уебсайтовете, в които влизате и приложенията, които използвате. Отнасяйте се с недоверие към съмнителни имейли с прикачени файлове и влизайте в сайтове, които са ви познати и притежават нужните защити, като активен SSL сертификат издаден от Сертифициран авторитетен SSL издател.

* Реализация на защита срещу атаката в проекта

**CookingHub** системата се хоства от безплатен сървър на Azure, поради което не мога да предоставя сигурна защита срещу Ransomware, но при евентуално заплащане и ъпгрейд на текущия план може да се вземат съответните мерки за защита.

*7. База данни (SQL injection)*

**7.1. SQL injection**

* Определение

SQL инжектирането е уязвимост в уеб защитата, която позволява на хакера да променя заявките, които дадено приложение изпраща към базата данни. Обикновено позволява на нападателя да преглежда данни, които обикновено не могат да бъдат извлечени. Това може да включва данни, принадлежащи на други потребители, или данни, които се използват от приложението. В много от случаите, нападателят може да модифицира, изтрива или променя данните, предизвиквайки постоянни промени в съдържанието и поведението на приложението. В някои ситуации, чрез SQL инжектиране нападателят може да компроментира основния сървър, друга back-end инфраструктура или дори да предизвика отказ на услугата.

* Кога една SQL injection атака е успешна?

Успешната SQL injection атака може да доведе до неоторизиран достъп на чувствителни данни от БД, като пароли, данни за кредитни карти или друга лична потребителска информация. Много от пропуските на уеб системите в днешни дни са в резултат на SQL injection атаки, водещи до щети на репутацията на дадената фирма и налагането на солидни глоби. В някои от случаите, нападателят може да намери уязвимост в системата (backdoor), което да доведе до дългосрочни проблеми и най-вече изтичане на данни за продължителен период от време.

* Примери

Нека разгледаме един уеб сайт за пазаруване, които показва продукти в различни категории. Когато потребителят кликне на категория “Подаръци”, браузърът изпраща следния URL:

**https://insecure-website.com/products?category=Gifts**

Тази заявка предизвиква приложението да изпрати следната SQL заявка за извличане на детайли за съответните продукти от БД:

**SELECT \* FROM products WHERE category = 'Gifts' AND released = 1**

Заявката изпраща следното към БД:

* Извлечи ми всичко (\*)
* От таблицата “продукти”
* Където категорията е “Подаръци”
* И released = 1 (тоест наличните подаръци)

“Released = 1” се използва, за да скрие продуктите от категорията, които не са налични. За липсващите продукти, можем да предположим, че “released = 0”. Приложението обаче не е изградило защита срещу SQL инжектиране и по този начин нападателят може да промени заявката си по следния начин:

**https://insecure-website.com/products?category=Gifts'--**

Съответно се изпраща следната заявка:

**SELECT \* FROM products WHERE category = 'Gifts'--' AND released = 1**

Ключовото тук е че “--” е оператор за коментар в SQL, което означава, че остатъкът от заявката се преобръща в коментар. По този начин се премахва частта “AND released = 1”, което означава, че ще бъдат показани всички продукти от категорията дори тези, които не са налични.

Отивайки още по надалеч, нападателят може да накара приложението да му покаже всички продукти без значение категорията, включително категориите, за които не знае, че съществуват със следната заявка:

**https://insecure-website.com/products?category=Gifts'+OR+1=1--**

Заявката връща всички продукти без значение категорията, защото 1=1 е винаги вярно.

* Как да се защитим?

1. Проверка на входните данни, които постъпват през формуляри, инпут полета или query параметри
2. Използване на параметризирани заявки
3. Използване на сторнати процедури (stored procedures)
4. Ескейпване на входните данни
5. Интегриране на WAF (Web application firewall)
6. Използване на ORM (Object Relational Mapping) фреймуърк, който автоматично транслира параметризирани SQL заявки

* Реализация на защита срещу атаката в проекта
* В проекта CookingHub се работи с последната версия на Entity Framework Core, който е ORM. Чрез него всички заявки са параметризирани и се предотвратяват проблеми свързани с SQL injection.

*8. Кражба на идентичност (Cross site request forgery)*

**8.1. Cross site request forgery**

* Определение

Cross site request forgery (CSRF) е атака, която принуждава крайния потребител да изпълни нежелани действия върху уеб приложението, в което е автентициран. С малко помощ от социалното инженерство (social engineering) като изпращане на връзка по имейл или чат, нападателят може да подмами потребителите на приложението да изпълнят определени действия. Ако жертвата е нормален потребител, успешната CSRF атака може да принуди потребителя да изпълни заявка за прехвърляне на средства, промяна на имейла и т.н. Ако жертвата пък е административният акаунт, CSRF може да компрометира цялото уеб приложение.

* Как работи атаката ?

За да бъде възможна CSRF атаката, трябва да са налице тези три ключови условия:

* **Съответно действие** – в приложението има действие (action), което нападателят има причина да предизвика. Това може да е привилегировано действие (като промяна на достъпа на другите потребители) или някакво действие върху специфични за потребителя данни (като промяна на собствената парола).
* **Cookie-based session handling** – включва издаване на една или повече HTTP заявки, като се знае, че приложението разчита единствено на бисквитките в сесията, за да се идентифицира конкретният потребител, който е направил request.
* **Няма непредсказуеми параметри на заявката** – Заявките, които изпълняват действието не съдържат никакви параметри, чиито стойности атакуващият не може да определи или отгатне. Например, когато потребителят си сменя паролата, функцията не е уязвима, ако нападателят е необходимо да знае стойността на вече съществуващата парола.

Нека дадем пример и да предположим следния сценарии:

Уеб приложението (за означение нека го кръстим като “A”) ни съдържа функция (действие), която позволява на потребителя да промени имейл адреса си в своя акаунт. Когато потребителят извърши това действие, той прави следната HTTP post заявка:

**POST /email/change HTTP/1.1  
Host: vulnerable-website.com  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 30  
Cookie: session=yvthwsztyeQkAPzeQ5gHgTvlyxHfsAfE**

**email=wiener@normal-user.com**

Тази заявка изпълнява описаните условия по-горе:

* Действието по промяна на имейл адреса в акаунта на потребителя представлява интерес за нападателя. След изпълнение на това действие, нападателят обикновено може да задейства нулиране на паролата и да поеме пълен контрол над акаунта на потребителя.
* Приложението използва сесийна бисквитка, за да идентифицира кой потребител е изпратил заявката. Няма други токъни или механизми за проследяване на потребителската сесия.
* Нападателят може лесно да определи стойностите на параметрите на заявката, които са необходими за извършване на атаката.

С тези изпълнени условия, нападателят може да си създаде проста уеб страница, съдържаща следния HTML:

**<html>  
  <body>  
    <form action="https://vulnerable-website.com/email/change" method="POST">  
      <input type="hidden" name="email" value=pwned@evil-user.net />  
    </form>  
    <script>  
      document.forms[0].submit();  
    </script>  
  </body>  
</html>**

Ако жертвата посети страницата на нападателя (уебсайт B), ще се случи следното:

* Страницата на нападателя (B) ще изпрати HTTP post заявка обратно към уязвимия уеб сайт (A)
* Ако жертвата е влязла в уязвимия уеб сайт (A), браузърът автоматично ще включи в заявката и сесиината бисквитка, необходима за идентификация (ако приемем, че SameSite не е включено)
* Уеб сайтът (A) ще обработи изпратената към него заявка от уебсайт B по нормален начин и ще третира тази заявка като направена от жертвата. Следователно, ще се промени имейл адресът на потребителя с този зададен от нападателя (**pwned@evil-user.net)**.
* Как да се защитим?

Най-надеждният начин за защита срещу CSRF атаки е да се включи CSRF токън към съответните заявки. Токънът трябва да бъде:

* Непредсказуем с висока ентропия
* Обвързан с потребителската сесия
* Строго валидиран във всеки един сценарий, преди да бъде изпълнено дадено действие (action)
* Реализация на защита срещу атаката в проекта
* Чрез използване на вградения филтър **AutoValidateAntiforgeryTokenAttribute**, там където се създават форми чрез тага <form> или има за въвеждане инпут поле, автоматично се добавя генериран токън, обвързан с текущата потребителска сесия, който е уникален и непредсказуем. При изпращане на дадена форма, този токън служи за валидация, че конкретно изпратените данни са от текущия потребител в сесията.

*9. Източници*

* [*https://evinat.com/bg/%D0%B1%D1%80%D1%83%D1%82-%D1%84%D0%BE%D1%80%D1%81-%D0%B0%D1%82%D0%B0%D0%BA%D0%B0-%D0%B4%D0%B5%D1%84%D0%B8%D0%BD%D0%B8%D1%86%D0%B8%D1%8F-%D0%B8-%D0%BA%D0%B0%D0%BA-%D0%B4%D0%B0-%D1%81%D0%B5-%D0%B7/*](https://evinat.com/bg/%D0%B1%D1%80%D1%83%D1%82-%D1%84%D0%BE%D1%80%D1%81-%D0%B0%D1%82%D0%B0%D0%BA%D0%B0-%D0%B4%D0%B5%D1%84%D0%B8%D0%BD%D0%B8%D1%86%D0%B8%D1%8F-%D0%B8-%D0%BA%D0%B0%D0%BA-%D0%B4%D0%B0-%D1%81%D0%B5-%D0%B7/)
* [*https://searchsecurity.techtarget.com/definition/dictionary-attack*](https://searchsecurity.techtarget.com/definition/dictionary-attack)
* [*https://www.internetsecurity.tips/what-is-a-dictionary-attack-and-how-to-prevent-it/*](https://www.internetsecurity.tips/what-is-a-dictionary-attack-and-how-to-prevent-it/)
* [*https://www.paessler.com/it-explained/packet-sniffing*](https://www.paessler.com/it-explained/packet-sniffing)
* [*https://freedomonline.bg/rechnik/man-in-the-middle/*](https://freedomonline.bg/rechnik/man-in-the-middle/)
* [*https://blog.logsign.com/10-steps-to-prevent-man-in-the-middle-attacks/*](https://blog.logsign.com/10-steps-to-prevent-man-in-the-middle-attacks/)
* [*https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/*](https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/)
* [*https://en.wikipedia.org/wiki/Slowloris\_(computer\_security)*](https://en.wikipedia.org/wiki/Slowloris_(computer_security))
* [*https://www.netscout.com/what-is-ddos/slowloris-attacks*](https://www.netscout.com/what-is-ddos/slowloris-attacks)
* [*https://owasp.org/www-community/attacks/xss/*](https://owasp.org/www-community/attacks/xss/)
* [*https://docs.microsoft.com/en-us/aspnet/core/security/cross-site-scripting?view=aspnetcore-5.0*](https://docs.microsoft.com/en-us/aspnet/core/security/cross-site-scripting?view=aspnetcore-5.0)
* [*https://www.icn.bg/bg/blog/friendly-geeks-saveti/ransomware-malware-notpetya/*](https://www.icn.bg/bg/blog/friendly-geeks-saveti/ransomware-malware-notpetya/)
* [*https://portswigger.net/web-security/sql-injection*](https://portswigger.net/web-security/sql-injection)
* [*https://www.hacksplaining.com/prevention/sql-injection*](https://www.hacksplaining.com/prevention/sql-injection)
* [*https://owasp.org/www-community/attacks/csrf*](https://owasp.org/www-community/attacks/csrf)

1. Линк към Github repository - <https://github.com/stanislavstoyanov99/CookingHub> [↑](#footnote-ref-1)
2. ASP.NET е технология за създаване на уеб приложения и уеб услуги, разработена от „Майкрософт“. [↑](#footnote-ref-2)
3. Cloudflare, Inc. е американска компания за уеб инфраструктура и защита на уебсайтове, предоставяща мрежови услуги за доставка на съдържание, смекчаване на DDoS, интернет сигурност и услуги за разпределени имена на домейни. [↑](#footnote-ref-3)