

# SOFTWARE SECURITY

Dott. Sabato Nocera | [snocera@unisa.it](mailto:snocera@unisa.it)

# DEFINIZIONI

- **Sicurezza informatica**: protezione delle informazioni e dei sistemi informativi per garantire:
  - ✓ **Riservatezza** (Confidentiality), ovvero l'accesso e la divulgazione soltanto a individui, entità e processi autorizzati;
  - ✓ **Integrità** (Integrity), cioè la protezione da modifiche e distruzioni improprie, nonché l'autenticità;
  - ✓ **Disponibilità** (Availability), per cui sono accessibili ed utilizzabili tempestivamente ed in modo affidabile.
  
- **Vulnerabilità**: debolezza in un sistema informativo, nelle procedure di sicurezza del sistema, nei controlli interni o nell'implementazione, che potrebbe essere sfruttata da un aggressore.

# OWASP TOP 10

La OWASP Top 10 è una classifica dei dieci più comuni e pericolosi rischi per la sicurezza delle applicazioni web.

Viene pubblicata annualmente dall'Open Web Application Security Project (OWASP) a supporto delle aziende e degli sviluppatori.



# A01:2021 – BROKEN ACCESS CONTROL

## Descrizione breve

Una stessa applicazione può essere utilizzata in modo diverso da utenti aventi ruoli differenti.

Il **controllo degli accessi** permette di verificare se un utente è autorizzato ad usufruire di una certa funzionalità dell'applicazione.

Rischi associati all'assenza di controlli degli accessi possono essere la divulgazione di informazioni non autorizzate, la modifica o la distruzione di tutti i dati.

## Codice vulnerabile

Il codice seguente utilizza dati non controllati in una query SQL per recuperare dal database le informazioni su un utente:

```
pstmt.setString(1,  
request.getParameter("acct"));  
  
ResultSet results =  
pstmt.executeQuery( );
```

## Scenario d'attacco

Supponendo che la query restituisca le informazioni associate al conto corrente avente come numero quello indicato dal parametro "acct", un malintenzionato potrebbe richiedere la visualizzazione di un conto corrente non proprio se non venisse verificato correttamente che il conto corrente a cui si sta tentando di accedere è il proprio:

```
https://example.com/app/accountInfo?acct=notmyacct
```

# A02:2021 – CRYPTOGRAPHIC FAILURES

## Descrizione breve

La **crittografia** può essere utilizzata per proteggere i dati manipolati dall'applicazione.

Data in input una stringa (es. password dell'utente), gli algoritmi di crittografia permettono di **codificarla** in un'altra stringa usando una **chiave crittografica**. La stringa ottenuta può essere memorizzata, così da essere successivamente recuperabile (es. login dell'utente) e **decodificabile** nella sua forma originaria.

## Codice vulnerabile

Il codice seguente legge una password da un file e la utilizza per connettersi a un database:

```
Properties prop = new
Properties();

prop.load(new
FileInputStream("config.proper
ties"));

String password =
Base64.decode(prop.getProperty
("password"));

DriverManager.getConnection(ur
l, usr, password);
```

## Scenario d'attacco

Un dipendente malintenzionato potrebbe scoprire:

- L'algoritmo utilizzato per crittografare la password per accedere al database, cioè **Base64**;
- La stringa codificata dall'algoritmo di crittografia.

Considerando che Base64 è un **algoritmo insicuro** per crittografare le password (la codifica e decodifica non usano chiavi crittografiche, quindi sono automatiche una volta conosciuto l'algoritmo), il malintenzionato potrebbe risalire alla password del database e compromettere il sistema.

# A03:2021 – INJECTION

## Descrizione breve

È possibile **iniettare codice sorgente eseguibile** in un'applicazione sfruttandone le falle di sicurezza, come l'assenza di controlli sui dati forniti in input dall'utente.

## Codice vulnerabile

Il codice seguente definisce una query attraverso la concatenazione di più stringhe, tra cui una fornita come parametro della richiesta HTTP:

```
String query = "SELECT * FROM  
accounts WHERE custID='" +  
request.getParameter("id") +  
"'";
```

## Scenario d'attacco

Immaginiamo che un malintenzionato fornisca come parametro in input la stringa:

```
mario-rossi'); DROP TABLE  
accounts;--
```

L'applicazione eseguirebbe prima una query al database per recuperare le informazioni dell'account con `custID='mario-rossi'`, successivamente sarebbe costretta ad eseguire una query che cancellerebbe la tabella `accounts`.

Questa tipologia di injection prende il nome di **SQL injection**.

# A04:2021 – INSECURE DESIGN

## Descrizione breve

**Progettare** un'applicazione trascurandone la sicurezza implica l'implementazione di una soluzione con numerose vulnerabilità, poiché, già dal principio, non sono stati **concepiti e previsti** le misure necessarie per difendersi da malintenzionati.

## Codice vulnerabile

Il codice seguente stampa le informazioni della traccia di esecuzione di un'eccezione in System.Err:

```
try {  
    /* ... */  
} catch(Exception e) {  
    e.printStackTrace();  
}
```

## Scenario d'attacco

Un attaccante potrebbe fornire un input che genera una certa eccezione (ad esempio relativa all'interrogazione del database).

Se l'eccezione non viene propriamente gestita nel blocco catch, l'output che l'attaccante visualizzerebbe potrebbe contenere informazioni sensibili (come la struttura della query SQL, quindi del database, o informazioni private).

# A05:2021 – SECURITY MISCONFIGURATION

## Descrizione breve

La sicurezza di un'applicazione dipende dall'**interazione** di varie componenti software: una loro **configurazione** scorretta potrebbe compromettere la sicurezza dell'intero sistema.

Ad esempio, l'applicazione potrebbe essere rilasciata con abilitate funzionalità di **debugging** e un attaccante potrebbe sfruttarle per ottenere informazioni sul sistema.

## Codice vulnerabile

Il codice seguente stampa le informazioni della traccia di esecuzione di un'eccezione in System.Err:

```
try {  
    /* ... */  
} catch(Exception e) {  
    e.printStackTrace();  
}
```

## Scenario d'attacco

Un attaccante potrebbe fornire un input che genera una certa eccezione (ad esempio relativa all'interrogazione del database).

Se l'eccezione non viene propriamente gestita nel blocco catch, l'output che l'attaccante visualizzerebbe potrebbe contenere informazioni sensibili (come la struttura della query SQL, quindi del database, o informazioni private).



# A06:2021 – VULNERABLE AND OUTDATED COMPONENTS

## Descrizione breve

Un'applicazione è costituita da molteplici componenti software e la sicurezza dei singoli determina la sicurezza dell'intero sistema.

Per questo motivo, è importante avere consapevolezza delle **versioni** che ne si utilizzano, sapere se presentano vulnerabilità, se sono ancora **supportate** o se sono disponibili **aggiornamenti**.

## Codice vulnerabile

Un server importa una versione delle librerie di Log4j in cui è stata trovata la vulnerabilità Log4Shell:

```
import org.apache.logging.log4j.LogManager;  
import org.apache.logging.log4j.Logger;  
  
import java.io.*;  
import java.sql.SQLException;  
import java.util.*;
```

## Scenario d'attacco

Un attaccante invia una richiesta al server contenente dati non attendibili che vengono poi processati da Log4j.

La versione utilizzata di Log4j non effettua controlli sulla bontà dei dati ricevuti ed innesca un meccanismo per l'iniezione di codice remoto.

# A07:2021 – IDENTIFICATION AND AUTHENTICATION FAILURES

## Descrizione breve

Rischi legati all'**autenticazione** dell'utente e alla gestione delle **sessioni** possono essere causati dall'utilizzo di password deboli, processi insicuri per il recupero delle credenziali e l'invalidazione scorretta delle sessioni.

## Codice vulnerabile

Il codice seguente utilizza una password codificata per connettersi a un database:

```
DriverManager.getConnection(url,  
"scott", "tiger");
```

## Scenario d'attacco

Un dipendente che ha accesso al codice sorgente può scoprire le credenziali per accedere in modo illecito al database. Inoltre, le stesse credenziali possono essere estratte dal bytecode dell'applicazione.

# A08:2021 – SOFTWARE AND DATA INTEGRITY FAILURES

## Descrizione breve

Un'applicazione può fare riferimento direttamente nel codice sorgente a componenti software e a dati provenienti da **fonti esterne**: è necessario verificarne l'integrità per evitare l'impiego di risorse malevole.

## Codice vulnerabile

Il codice seguente incorpora un frammento di codice JavaScript da un content delivery network (CDN):

```
<script  
src="https://cdnexample.com/script.js">
```

## Scenario d'attacco

Un malintenzionato potrebbe corrompere il codice presente in <https://cdnexample.com/script.js>, cosicché, quando questo viene incorporato lato client, possa essere iniettato codice malevolo nell'applicazione.

# A09:2021 – SECURITY LOGGING AND MONITORING FAILURES

## Descrizione breve

Il **monitoraggio** delle attività che coinvolgono un'applicazione può essere effettuato attraverso la stampa di **messaggi di log**. Un loro corretto utilizzo può permettere di identificare prontamente errori e vulnerabilità del sistema (es. messaggi di errore, query eseguite al database, continui tentativi di login falliti).

## Codice vulnerabile

Il codice seguente riporta un tentativo di autenticazione di un utente:

```
if LoginUser(){  
    // Login successful  
    RunProgram();  
} else {  
    // Login unsuccessful  
    LoginRetry();  
}
```

## Scenario d'attacco

Se un tentativo di login fallisce, è possibile effettuare un nuovo tentativo. Un malintenzionato potrebbe provare e riprovare ad accedere al sistema cercando di indovinare le credenziali e, sfortunatamente, i tentativi falliti di login non verrebbero registrati, quindi nessuno si accorgerebbe dell'attacco subito.

# A10:2021 – SERVER-SIDE REQUEST FORGERY

## Descrizione breve

Il rischio di Server-Side Request Forgery (SSRF) si verifica ogni volta che un'applicazione web recupera una risorsa remota senza validare l'**URL** fornito dall'utente. Questo potrebbe permettere ad un malintenzionato di forzare l'applicazione ad inviare una richiesta malevola ad una **destinazione inattesa**, anche quando quest'ultima è protetta da un firewall o una VPN.

## Codice vulnerabile

Il codice seguente riporta un tentativo di accesso ad un sito web:

```
URL url = new  
URL(req.getParameter("url"));  
  
URLConnection conn =  
(URLConnection)  
url.openConnection();
```

## Scenario d'attacco

Consideriamo un'applicazione web che interroga un server per recuperare dei dati attraverso delle API. Per accedere a tali API è necessario specificare un URL (parametro "**url**").

Un malintenzionato potrebbe indicare **http://localhost/admin** come URL, per provare ad accedere al contenuto della cartella **admin** presente sullo stesso server dell'applicazione. Dato che la richiesta d'accesso sarebbe effettuata localmente (**localhost**), le normali misure di sicurezza verrebbero eluse.

# APPLICATION SECURITY TESTING

È possibile individuare e risolvere potenziali vulnerabilità nelle applicazioni impiegando diversi approcci e strumenti.

	Analisi statica: <i>Static Application Security Testing</i>	Analisi dinamica: <i>Dynamic Application Security Testing</i>
Funzionamento	Il <b>codice</b> dell'applicazione viene scansionato alla ricerca di linee di codice potenzialmente vulnerabili	L'applicazione in <b>esecuzione</b> viene esaminata alla ricerca di comportamenti insicuri
Prospettiva	Sviluppatore: conosce la progettazione e l'implementazione dell'applicazione	Hacker: necessita di acquisire prima le informazioni sull'applicazione per poi violarla
Visibilità	Le vulnerabilità identificate vengono evidenziate nel codice (con possibili suggerimenti per la loro risoluzione)	Necessità di impiegare (molto) tempo per ricercare le vulnerabilità identificate all'interno del codice
Applicabilità	Necessitando soltanto del codice sorgente, si può svolgere dal momento in cui si inizia l'implementazione	Necessitando dell'applicazione in esecuzione, si può svolgere soltanto quando il codice compila
Falsi positivi	Molte delle vulnerabilità identificate possono non costituire un pericolo per la sicurezza dell'applicazione quando questa in esecuzione viene utilizzata	Difficilmente una vulnerabilità identificata non rappresenta un pericolo per la sicurezza del sistema

# FUNZIONAMENTO DI SONARCLOUD

## Regole

SonarCloud definisce un insieme di regole che, al momento della scansione del codice sorgente, verificano l'aderenza di quest'ultimo ai buoni principi di programmazione, tra cui la sicurezza.

Tali regole sono state create sulla base di standard di sicurezza, come OWASP Top 10.

Quando un frammento di codice viola una regola, viene sollevata una issue.

## Issue

Le issue relative alla sicurezza possono essere di due diversi tipi:

- **Vulnerability**, che richiedono di essere immediatamente risolte perché rappresentative di una vulnerabilità del codice;
- **Security hotspot**, per cui è necessario verificarne la pericolosità per decidere se risolverle oppure no.

## Severity

A ciascuna issue viene assegnata una severity, ovvero un grado di rischio associato alla probabilità che venga sfruttata da un malintenzionato e all'impatto che avrebbe.

Dalla severity minore alla maggiore troviamo:

- Info
- Minor
- Major
- Critical
- Blocker

## DELIVERING CODE IN PRODUCTION WITH DEBUG FEATURES ACTIVATED

Le funzioni di **debug** di un'applicazione consentono agli sviluppatori di trovare più facilmente i problemi attraverso il monitoraggio del comportamento del sistema. Un aggressore potrebbe sfruttare tali informazioni per ottenere informazioni sensibili sul sistema ed i suoi utenti, per poi attaccarlo.

Ad esempio, ciò può avvenire a causa della funziona `printStackTrace()`, la quale stampa un Throwable e la sua traccia di stack in System.Err. Per stampare le informazioni dei Throwable è consigliabile usare i **logger**.

### Security Hotspot

```
try {  
    /* ... */  
} catch(Exception e) {  
    e.printStackTrace(); // Sensitive  
}
```

### Risoluzione

```
try {  
    /* ... */  
} catch(Exception e) {  
    LOGGER.log("context", e);  
}
```



## FORMATTING SQL QUERIES

Le query SQL formattate attraverso la **concatenazione di più stringhe** possono portare ad SQL injection.

Quando una query è costruita attraverso la concatenazione di valori inseriti in input dagli utenti (es. campi di un form), è necessario disporre query parametriche attraverso **PreparedStatement** e "caricare" i relativi parametri.

### Security Hotspot

```
Statement stmt2 = con.createStatement();  
ResultSet rs2 = stmt2.executeQuery("select FNAME, LNAME, SSN  
" + "from USERS where UNAME=" + user); // Sensitive  
  
PreparedStatement pstmt = con.prepareStatement("select  
FNAME, LNAME, SSN " + "from USERS where UNAME=" + user); //  
Sensitive  
  
ResultSet rs3 = pstmt.executeQuery();
```

### Risoluzione

```
Statement stmt1 = con.createStatement();  
ResultSet rs1 = stmt1.executeQuery("select FNAME, LNAME,  
SSN" + "from USERS"); // No issue; hardcoded query  
  
String query = "select FNAME, LNAME, SSN from USERS where  
UNAME=?"  
  
PreparedStatement pstmt = con.prepareStatement(query);  
  
pstmt.setString(1, user); // Good; PreparedStatement escape  
their inputs.  
  
ResultSet rs2 = pstmt.executeQuery();
```

## DISABLING RESOURCE INTEGRITY FEATURES

L'acquisizione di **risorse esterne** senza verificarne l'integrità può compromettere la sicurezza di un'applicazione se la fonte da cui proviene viene compromessa. È possibile verificare l'integrità di una risorsa esterna nel seguente modo:

1. Nel codice sorgente, la risorsa viene "etichettata" con una stringa alfanumerica (*digest*) generata dalla codifica del suo contenuto;
2. Quando un'applicazione accede alla risorsa esterna, viene calcolato il *digest* sulla base del contenuto della risorsa a cui si sta tentando di accedere;
3. Se il *digest* con cui la risorsa è stata "etichettata" è uguale a quello calcolato al momento in cui si sta tentando l'accesso, allora significa che la risorsa che si sta recuperando è quella che ci si aspetta (integra), altrimenti è malevola.

### Security Hotspot

```
<script src="https://cdnexample.com/script.js"></script>  
<!-- Sensitive -->
```

### Risoluzione

```
<script src="https://cdnexample.com/script.js"  
  integrity="sha384-  
oqVuAfXRRKap7fdgcCY5uykM6+R9GqQ8K/uxy9rx7HNQlGYl1kPzQho1wx4Jw  
Y8wC"></script>  
  
<!-- Compliant: integrity value should be replaced with the  
digest of the expected resource -->
```

## USING SLOW REGULAR EXPRESSIONS

Le espressioni regolari permettono di **controllare la validità degli input** inseriti dagli utenti. Quando la stringa in input risulta essere molto lunga o l'espressione regolare molto complessa, la computazione può richiedere tantissimo tempo: inviando molteplici richieste HTTP in breve tempo, un attaccante potrebbe causare un denial of service dell'applicazione.

Per prevenire tali situazioni, è consigliabile utilizzare espressioni regolari minimali (perlomeno senza inutili ripetizioni), definire una lunghezza massima per l'input fornito dall'utente o impostare dei time-out oltre i quali la computazione termina.

### Security Hotspot

```
import re
```

```
pattern = "(a+)+"
```

# Questo pattern causerà un blocco infinito quando utilizzato con una stringa lunga contenente molti caratteri "a" consecutivi

```
re.match(pattern, "a" * 1000000)
```

### Risoluzione

```
import re
```

```
pattern = "a+"
```

# Questo pattern non causerà problemi di prestazioni con una stringa lunga contenente molti caratteri "a" consecutivi

```
re.match(pattern, "a" * 1000000)
```

## HARD-CODED PASSWORDS

Poiché è facile **estrarre stringhe** dal codice sorgente o dal codice binario di un'applicazione, le password non dovrebbero essere codificate in modo rigido, cioè direttamente nel codice. Tale problematica risulta particolarmente rilevante per le applicazioni distribuite o open source.

È consigliabile che le password siano memorizzate al di fuori del codice in un file di configurazione, in un database o in un servizio di gestione delle password.

### Security Hotspot

```
String username = "steve";  
String password = "blue";  
  
Connection conn =  
    DriverManager.getConnection("jdbc:mysql://localhost/test?" +  
        "user=" + uname + "&password=" + password);  
  
// Sensitive
```

### Risoluzione

```
String username = getEncryptedUser();  
String password = getEncryptedPassword();  
  
Connection conn =  
    DriverManager.getConnection("jdbc:mysql://localhost/test?" +  
        "user=" + uname + "&password=" + password);
```

## USING PSEUDORANDOM NUMBER GENERATORS

I generatori di numeri pseudo-casuali (PRNG, dall'inglese pseudo-random number generator) sono algoritmi **deterministici** che permettono di creare sequenze numeriche **prevedibili**; utilizzarli in contesti in cui è richiesta **imprevedibilità** può costituire una minaccia per il sistema perché un attaccante potrebbe indovinare e sfruttare i numeri generati, per questo motivo è bene preferire l'utilizzo di generatori di numeri casuali (RNG, dall'inglese number generator) crittograficamente forti.

### Security Hotspot

```
Random random = new Random();  
// Sensitive use of Random  
byte bytes[] = new byte[20];  
random.nextBytes(bytes);  
// Check if bytes is used for hashing, encryption, etc...
```

### Risoluzione

```
SecureRandom random = new SecureRandom();  
// Compliant for security-sensitive use cases  
byte bytes[] = new byte[20];  
random.nextBytes(bytes);
```

## AUTHORIZING AN OPENED WINDOW TO ACCESS BACK TO THE ORIGINATING WINDOW

Il tag `<a href=>` viene utilizzato per collegare una pagina web X ad una pagina web Y. Utilizzando l'attributo `target="_blank"`, quando l'utente clicca sul collegamento ipertestuale nella pagina web X, la pagina web Y verrà aperta in un nuova scheda del browser (quindi la pagina web X rimarrà ancora **aperta**).

Tale comportamento può costituire una minaccia alla sicurezza del sistema perché la pagina web Y potrebbe **accedere** e modificare la pagina web X, ad esempio cambiandola in una pagina web Z malevola che chiede all'utente l'inserimento di dati sensibili. Per evitare ciò, è possibile utilizzare l'attributo `rel=noopener`.

### Security Hotspot

```
<a href="http://example.com/dangerous" target="_blank">
<!-- Sensitive -->
```

```
<a href="{{variable}}" target="_blank">
<!-- Sensitive -->
```

### Risoluzione

```
<a href="http://petsocialnetwork.io" target="_blank"
rel="noopener"> <!-- Compliant -->
```

## USING CLEAR-TEXT PROTOCOLS

L'utilizzo di protocolli per lo scambio di informazioni che non ne prevedono la crittografia durante il **trasporto** può permettere a malintenzionati di leggerne e modificarne il contenuto, in quanto il traffico di rete può essere **intercettato**. È possibile evitare tali pericoli impiegando protocolli sicuri:

- **telnet** può essere sostituito con **ssh** ;
- **ftp** può essere sostituito con **sftp**, **scp**, **ftps**;
- **http** può essere sostituito con **https**.

## Security Hotspot

```
ConnectionSpec spec = new  
ConnectionSpec.Builder(ConnectionSpec.CLEARTEXT).build();  
  
// Sensitive
```

## Risoluzione

```
ConnectionSpec spec = new  
ConnectionSpec.Builder(ConnectionSpec.MODERN_TLS).build();  
  
// Compliant
```

## DYNAMICALLY EXECUTING CODE

L'esecuzione del codice in modo **dinamico** (ad esempio sulla base degli input forniti da utenti) può costituire un pericolo di injection sia lato client che lato server.

È consigliato non eseguire mai codice sconosciuto proveniente da fonti non attendibili

### Security Hotspot

```
var greeting = "good morning"
function speak(str) {
  eval(str) // Sensitive
  console.log(greeting)
}
speak("var greeting = 'meow'")
```

### Risoluzione

```
var morning = "good morning"
function speak(greeting) {
  console.log(morning)
}
speak(morning)
// Compliant
```



## ENCRYPTION ALGORITHMS SHOULD BE USED WITH SECURE MODE AND PADDING SCHEME

Gli algoritmi di crittografia dovrebbero utilizzare **modalità sicure** e **schemi di padding** dove appropriati per garantire la confidenzialità e l'integrità dei dati.

Per gli algoritmi di crittografia a blocchi (come AES), bisognerebbe prevedere l'utilizzo della modalità **GCM** al posto di **ECB** e **CBC**; invece, per l'algoritmo di crittografia RSA è indicato lo schema di padding **OAEP**.

### Vulnerability

```
Cipher.getInstance("AES"); // Noncompliant: by default ECB mode is chosen
```

```
Cipher.getInstance("AES/ECB/NoPadding"); // Noncompliant: ECB doesn't provide serious message confidentiality
```

```
Cipher.getInstance("AES/CBC/PKCS5Padding"); // Noncompliant: Vulnerable to Padding Oracle attacks
```

```
Cipher.getInstance("RSA/None/NoPadding"); // Noncompliant: RSA without OAEP padding scheme is not recommended
```

### Risoluzione

```
Cipher.getInstance("AES/GCM/NoPadding");
```

```
Cipher.getInstance("RSA/None/OAEPWITHSHA-256ANDMGF1PADDING"); // or the ECB mode can be used for RSA when "None" is not available with the security provider used - in that case, ECB will be treated as "None" for RSA.
```

```
Cipher.getInstance("RSA/ECB/OAEPWITHSHA-256ANDMGF1PADDING");
```

## CIPHER ALGORITHMS SHOULD BE ROBUST

Gli algoritmi di cifratura forti sono resistenti alla crittoanalisi e alle più note tipologie di attacchi, come quelli a forza bruta.

Una raccomandazione generale è quella di utilizzare solo algoritmi di cifratura **intensamente testati e promossi dalla comunità crittografica**.

### Vulnerability

```
Cipher c1 = Cipher.getInstance("DES");  
// Noncompliant: DES works with 56-bit keys allow attacks  
via exhaustive search
```

### Risoluzione

```
Cipher c31 = Cipher.getInstance("AES/GCM/NoPadding");  
// Compliant
```

## USING WEAK HASHING ALGORITHMS

Algoritmi di **hash crittografici** (es. MD5, SHA-1) non sono più considerati sicuri, perché basta un piccolo sforzo computazionale per trovare due o più input diversi che producono lo stesso hash. Quando è necessario garantire elevati standard di sicurezza, è consigliato utilizzare algoritmi più sicuri per effettuare l'hashing (es. SHA-256, SHA-512).

### Security Hotspot

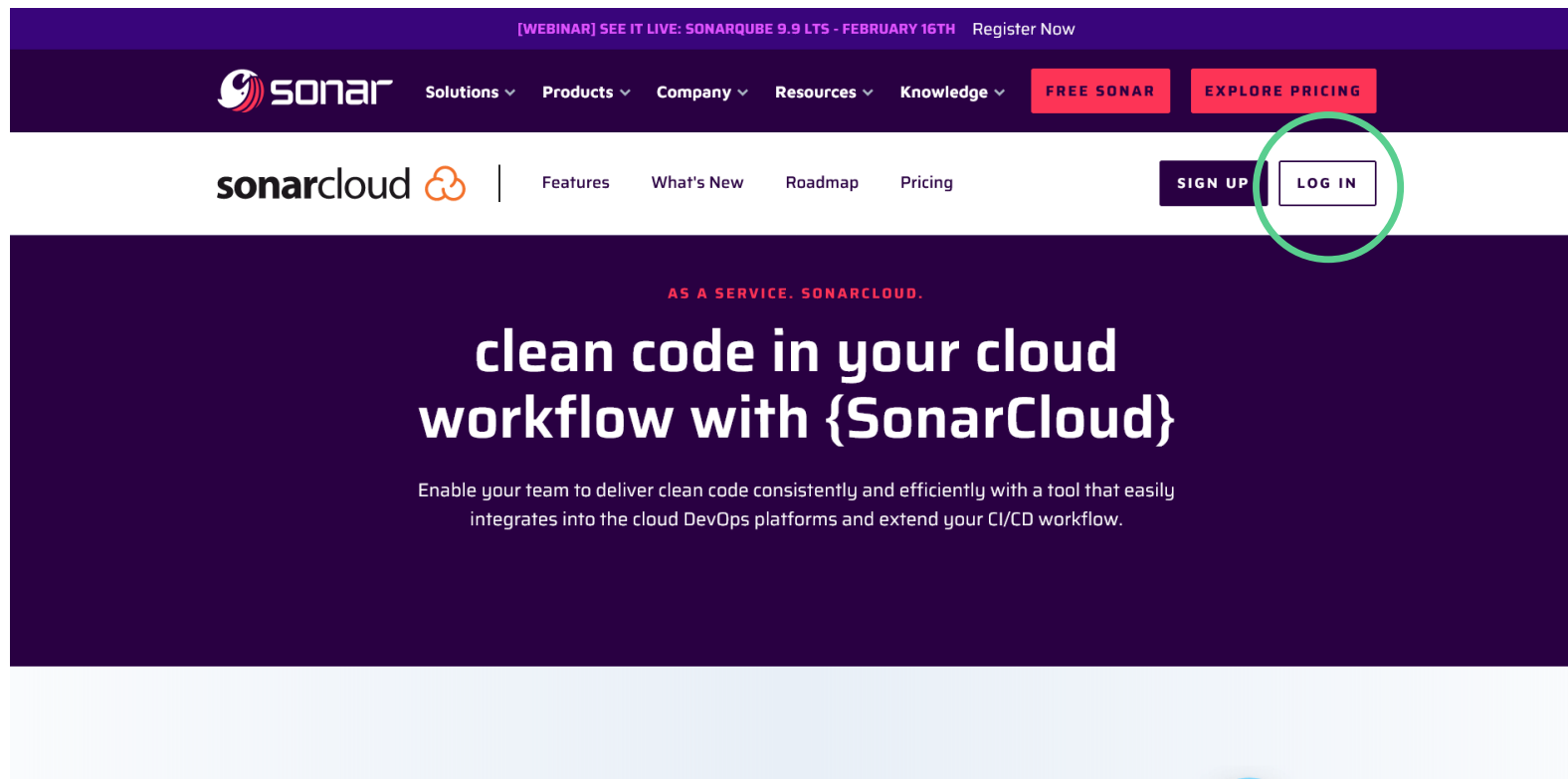
```
MessageDigest md1 = MessageDigest.getInstance("SHA");  
// Sensitive: SHA is not a standard name, for most security  
providers it's an alias of SHA-1
```

```
MessageDigest md2 = MessageDigest.getInstance("SHA1");  
// Sensitive
```

### Risoluzione


```
MessageDigest md1 = MessageDigest.getInstance("SHA-512");  
// Compliant
```


# SONARCLOUD - HOMEPAGE



The screenshot shows the SonarCloud homepage. At the top, a purple banner contains the text "[WEBINAR] SEE IT LIVE: SONARQUBE 9.9 LTS - FEBRUARY 16TH" and a "Register Now" link. Below this is a navigation bar with the Sonar logo, links for "Solutions", "Products", "Company", "Resources", and "Knowledge", and two red buttons: "FREE SONAR" and "EXPLORE PRICING". The main navigation bar features the "sonarcloud" logo with a cloud icon, followed by links for "Features", "What's New", "Roadmap", and "Pricing". On the right side of this bar are "SIGN UP" and "LOG IN" buttons, with the "LOG IN" button highlighted by a green circle. The main content area has a dark purple background with the text "AS A SERVICE. SONARCLOUD." in red, followed by the headline "clean code in your cloud workflow with {SonarCloud}" in white. Below the headline, a paragraph states: "Enable your team to deliver clean code consistently and efficiently with a tool that easily integrates into the cloud DevOps platforms and extend your CI/CD workflow."

[WEBINAR] SEE IT LIVE: SONARQUBE 9.9 LTS - FEBRUARY 16TH Register Now

 Solutions ▾ Products ▾ Company ▾ Resources ▾ Knowledge ▾ FREE SONAR EXPLORE PRICING

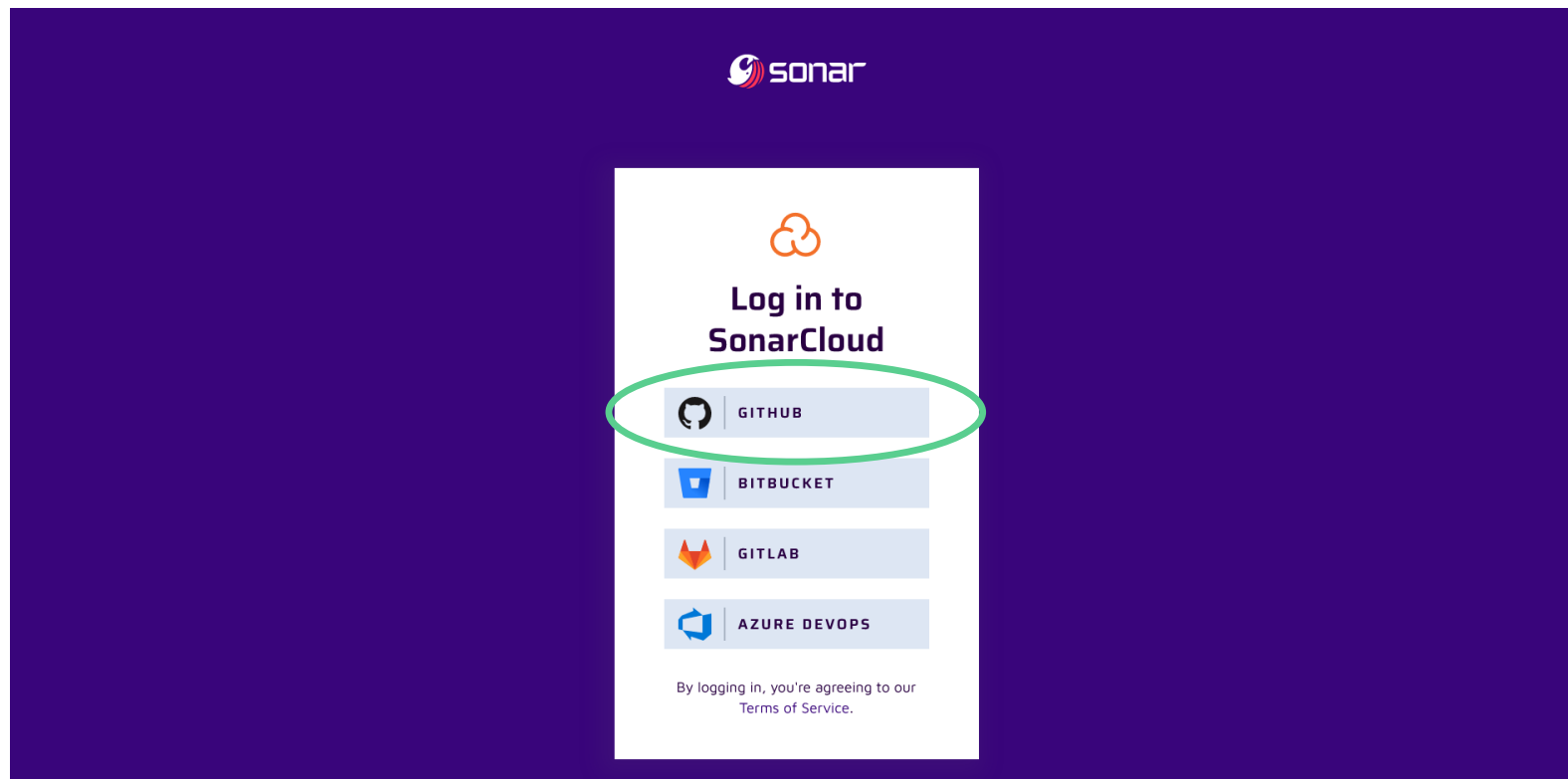
sonarcloud  | Features What's New Roadmap Pricing SIGN UP LOG IN

AS A SERVICE. SONARCLOUD.

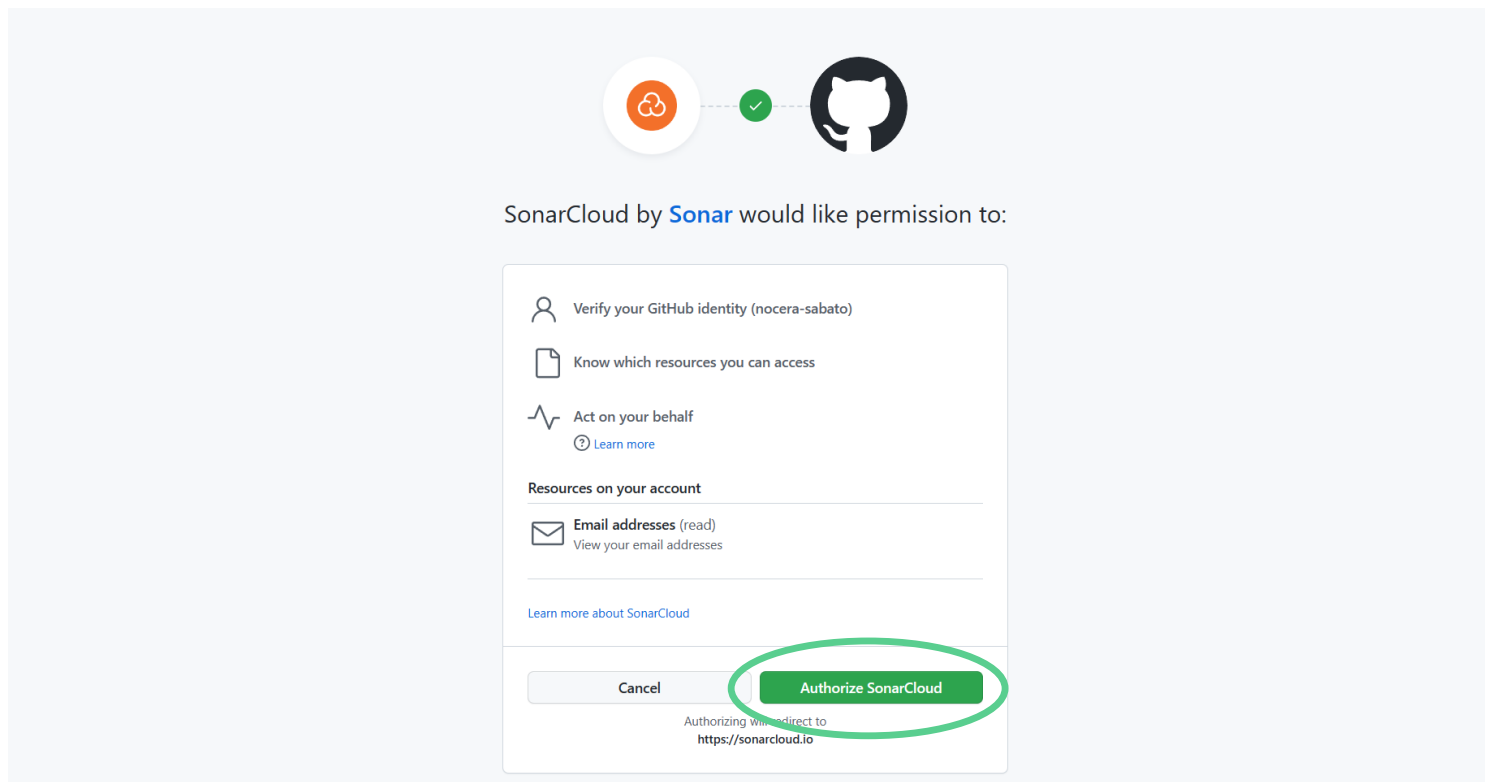
## clean code in your cloud workflow with {SonarCloud}

Enable your team to deliver clean code consistently and efficiently with a tool that easily integrates into the cloud DevOps platforms and extend your CI/CD workflow.

# SONARCLOUD - LOGIN



# SONARCLOUD - LOGIN



# SONARCLOUD - WELCOME PAGE



## Welcome to SonarCloud

Let us help you get started in your journey to code quality



### Analyze your first projects

Grant access to the SonarCloud application in your GitHub organization or user account. You will then be able to select which repositories you want to analyze.

 [Import an organization from GitHub](#)

Just testing? You can create projects manually.




### Join an organization

Now that you have an account on SonarCloud, just ask the Organization Administrator to add you manually.

[Learn More](#) 

# SONARCLOUD - INSTALL SONARCLOUD



## Install SonarCloud

Install on your personal account Sabato Nocera

☒ **All repositories**  
This applies to all current *and* future repositories owned by the resource owner.  
Also includes public repositories (read-only).

☐ **Only select repositories**  
Select at least one repository.  
Also includes public repositories (read-only).

with these permissions:

- ✓ **Read** access to code and metadata
- ✓ **Read and write** access to checks, commit statuses, pull requests, and security events

User permissions  
SonarCloud can also request users' permission to the following resources. These permissions will be requested and authorized on an individual-user basis.

- ✓ **Read** access to email addresses

**Install** Cancel



# SONARCLOUD – CREATE ORGANIZATION

## Create an organization

An organization is a space where a team or a whole company can collaborate across many projects.

### 1 Import organization details

Import  **Sabato Nocera** into a SonarCloud organization [×](#)

Name


Up to 255 characters

Key \* 

Organization key must start with a lowercase letter or number, followed by lowercase letters, numbers or hyphens, and must end with a letter or number. Maximum length: 255 characters.

[Add additional info](#) 



All members from your GitHub organization Sabato Nocera will be added to your SonarCloud organization. As they connect to SonarCloud with their GitHub account, members will automatically have access to your SonarCloud organization and its projects. [See all members on GitHub](#) 

Continue

# SONARCLOUD – CREATE ORGANIZATION

## Create an organization

An organization is a space where a team or a whole company can collaborate across many projects.

1 Import organization details

✓ nocera-sabato

2 Choose a plan

☐ Paid plan from 10 €

- ✓ Unlimited private projects
- ✓ Strict control over who can view your private data
- ✓ No commitments, cancel anytime
- ✓ 14 days free trial.

[Learn more](#)

✓ Recommended because your organization contains private repositories

☒ Free plan 0 €

All projects you analyze will be public.  
Anyone can browse source code.



SonarCloud won't allow your private repositories to be imported from GitHub if you choose this plan.

Create Organization

Bisogna avere almeno un repository pubblico su GitHub!


# SONARCLOUD – CREATE PROJECT


Analyze projects - Select repositories

Organization \*

[Import another organization](#)

☐ Select all available repositories



☐  prova\_1

Analyze private projects with our Paid Plan from 10 €

- ✓ Unlimited private projects
- ✓ Strict control over who can view your private data
- ✓ No commitments, cancel anytime
- ✓ 14 days free trial.



[Learn more](#)

Upgrade


# SONARCLOUD – CREATE PROJECT


Analyze projects - Select repositories

Organization \*

 Sabato Nocera nocera-sabato  [Import another organization](#)

☒ Select all available repositories



☒  prova\_1


**1 repository selected**

1 repository will be created as a public project on SonarCloud

**Set Up**

Analyze private projects with our Paid Plan from 10 €

- ✓ Unlimited private projects
- ✓ Strict control over who can view your private data
- ✓ No commitments, cancel anytime
- ✓ 14 days free trial.

[Learn more](#) 

Upgrade

# SONARCLOUD – PROJECT

The screenshot displays the SonarCloud interface for configuring a project named 'prova\_1'. The top navigation bar includes 'My Projects', 'My Issues', 'Explore', and a search icon. The breadcrumb trail shows 'Sabato Nocera > prova\_1 > Configure'. The left sidebar, under the 'sonarcloud' logo, shows the project name 'prova\_1' with 'PUBLIC' status and icons for a fork and star. Below this, the 'Configure' section is active, showing 'Main Branch', 'Pull Requests' (0), and 'Branches' (0). Further down are 'Information' and 'Administration' sections, and a 'Collapse' button at the bottom. The main content area is titled 'Choose your Analysis Method' and offers five options: 'With GitHub Actions' (circled in green), 'With Travis CI', 'With CircleCI', 'With other CI tools', and 'Manually'. The 'With other CI tools' option includes a note: 'SonarCloud integrates with your workflow no matter which CI tool you're using.' The 'Manually' option states: 'Use this for testing. Other modes are recommended to help you set up your CI environment.' The footer contains copyright information '© 2008-2023, SonarCloud by SonarSource SA. All rights reserved.' and a series of links: 'Terms', 'Pricing', 'Privacy', 'Security', 'Community', 'Documentation', 'Contact us', 'Status', and 'About'.

sonarcloud

prova\_1

PUBLIC

Configure

Main Branch

Pull Requests 0

Branches 0

Information

Administration

Collapse

My Projects My Issues Explore

Sabato Nocera > prova\_1 > Configure

### Choose your Analysis Method

[With GitHub Actions](#)

[With Travis CI](#)

[With CircleCI](#)

[With other CI tools](#)

SonarCloud integrates with your workflow no matter which CI tool you're using.

[Manually](#)

Use this for testing. Other modes are recommended to help you set up your CI environment.

© 2008-2023, SonarCloud by SonarSource SA. All rights reserved.

[Terms](#) [Pricing](#) [Privacy](#) [Security](#) [Community](#) [Documentation](#) [Contact us](#) [Status](#) [About](#)

# SONARCLOUD – PROJECT

sonarcloud

prova\_1

PUBLIC

Configure

Main Branch

Pull Requests 0

Branches 0

Information

Administration

Collapse

My Projects My Issues Explore Q

Sabato Nocera > prova\_1 > Configure > With GitHub Actions

## Analyze a project with a GitHub Action

### Create a GitHub Secret

In your GitHub repository, go to [Settings > Secrets > Actions](#) and create a new secret with the following details:

- 1 In the **Name** field, enter `SONAR_TOKEN`
- 2 In the **Value** field, enter `a7f35b40b426fb4b083192dc3326c4098c51fa06`

### Create or update a build file

What option best describes your build?

Maven Gradle C, C++ or ObjC .NET Other (for JS, TS, Go, Python, PHP, ...)

© 2008-2023, SonarCloud by SonarSource SA. All rights reserved.

[Terms](#) [Pricing](#) [Privacy](#) [Security](#) [Community](#) [Documentation](#) [Contact us](#) [Status](#) [About](#)

# SONARCLOUD – PROJECT

nocera-sabato / prova\_1 Public Pin Unwatch 1 Fork 0 Star 0

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

[General](#) [Actions secrets / New secret](#)

**Access**

- Collaborators
- Moderation options

**Code and automation**

- Branches
- Tags
- Actions
- Webhooks
- Environments
- Codespaces
- Pages

**Security**

- Code security and analysis
- Deploy keys
- Secrets and variables**

**Name \***

SONAR\_TOKEN

**Secret \***

a7f35b40b426fb4b083192dc3326c4098c51fa06

**Add secret**

# SONARCLOUD – PROJECT

The screenshot displays the SonarCloud web interface for configuring a project named 'prova\_1'. The left sidebar contains navigation links: 'Configure', 'Main Branch', 'Pull Requests', 'Branches', 'Information', 'Administration', and 'Collapse'. The main content area is titled 'Analyze a project with a GitHub Action' and includes a section 'Create a GitHub Secret' with instructions to create a secret in a GitHub repository. Below this, there are two numbered steps: 1. In the Name field, enter 'SONAR\_TOKEN'. 2. In the Value field, enter a long alphanumeric string. Further down, the 'Create or update a build file' section asks 'What option best describes your build?' and provides several buttons: 'Maven', 'Gradle', 'C, C++ or ObjC', '.NET', and 'Other (for JS, TS, Go, Python, PHP, ...)'. The 'Other' button is circled in green. The footer contains copyright information and links to Terms, Pricing, Privacy, Security, Community, Documentation, Contact us, Status, and About.

sonarcloud

prova\_1

PUBLIC

Configure

Main Branch

Pull Requests 0

Branches 0

Information

Administration

Collapse

My Projects My Issues Explore

Sabato Nocera > prova\_1 > Configure > With GitHub Actions

### Analyze a project with a GitHub Action

#### Create a GitHub Secret

In your GitHub repository, go to [Settings > Secrets > Actions](#) and create a new secret with the following details:

- 1 In the **Name** field, enter `SONAR_TOKEN`
- 2 In the **Value** field, enter `a7f35b40b426fb4b083192dc3326c4098c51fa06`

#### Create or update a build file

What option best describes your build?

Maven Gradle C, C++ or ObjC .NET **Other (for JS, TS, Go, Python, PHP, ...)**

© 2008-2023, SonarCloud by SonarSource SA. All rights reserved.

[Terms](#) [Pricing](#) [Privacy](#) [Security](#) [Community](#) [Documentation](#) [Contact us](#) [Status](#) [About](#)



# SONARCLOUD – PROJECT

**sonarcloud**

prova\_1  
PUBLIC

Configure

Main Branch

Pull Requests 0

Branches 0

Information

Administration

Collapse

My Projects My Issues Explore

Sabato Nocera > prova\_1 > Configure > With GitHub Actions

What option best describes your build?

Maven Gradle **C, C++ or ObjC** .NET Other (for JS, TS, Go, Python, PHP, ...)

Create or update your `.github/workflows/build.yml`

Here is a base configuration to run a SonarCloud analysis on your master branch and Pull Requests. If you already have some GitHub Actions, you might want to just add some of these new steps to an existing one.

```
name: Build
on:
  push:
    branches:
      - main
  pull_request:
    types: [opened, synchronize, reopened]
jobs:
  sonarcloud:
    name: SonarCloud
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
        with:
          fetch-depth: 0 # Shallow clones should be disabled for a better relevancy of analysis
      - name: SonarCloud Scan
        uses: SonarSource/sonarcloud-github-action@master
        env:
          GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN } # Needed to get PR information, if any
          SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
```

Copy

# SONARCLOUD – PROJECT

The screenshot shows the GitHub interface for the repository 'nocera-sabato / prova\_1'. The repository is public and has 1 commit. The 'Add file' button is circled in green, and its dropdown menu is open, showing 'Create new file' (also circled in green) and 'Upload files'. The repository contains a file named 'SQLInjectionExample.java' created 20 minutes ago. The right sidebar shows the repository's metadata: no description, 0 stars, 1 watching, and 0 forks. The 'Releases' and 'Packages' sections indicate no published content. The 'Languages' section shows 100.0% Java.

nocera-sabato / prova\_1 Public

Pin Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

Go to file Add file Code

Create new file Upload files 1 commit

nocera-sabato Create SQLInjectionExample.java

SQLInjectionExample.java Create SQLInjectionExample.java 20 minutes ago

Help people interested in this repository understand your project by adding a README. [Add a README](#)

About

No description, website, or topics provided.

0 stars

1 watching

0 forks

Releases

No releases published  
[Create a new release](#)

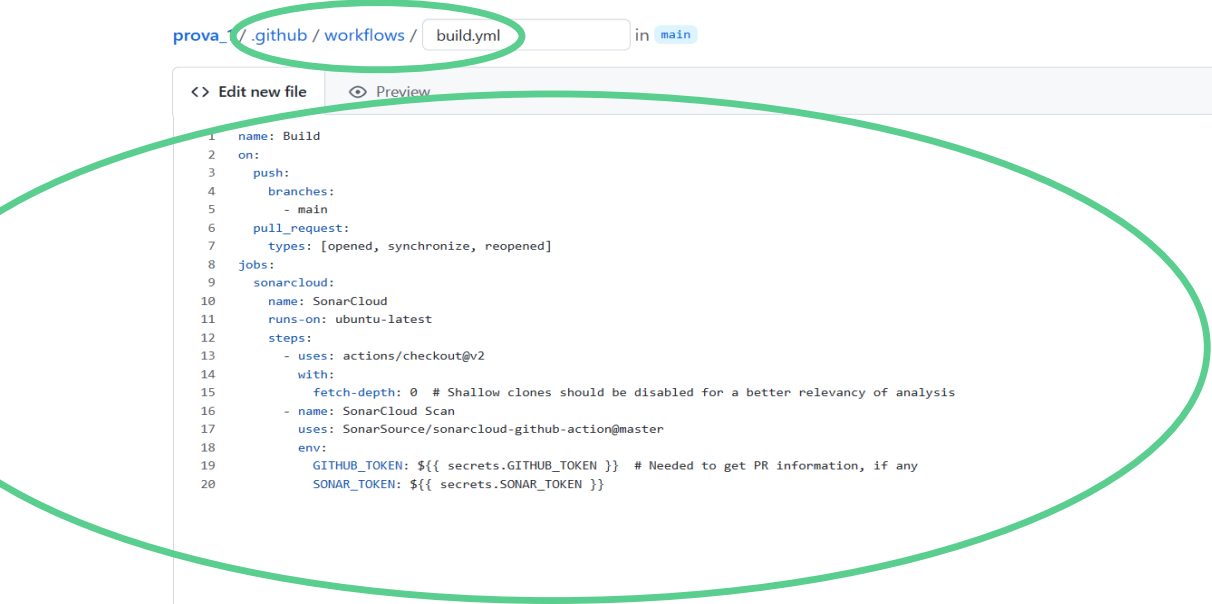
Packages

No packages published  
[Publish your first package](#)

Languages

Java 100.0%

# SONARCLOUD – PROJECT



The screenshot displays the GitHub interface for a repository named 'prova\_1' by user 'nocera-sabato'. The file path '.github / workflows / build.yml' is highlighted with a green circle. The file content, which is a GitHub Actions workflow, is enclosed in a large green oval. The workflow is configured to run on the 'main' branch and includes steps for checking out the code and performing a SonarCloud scan. The scan step uses the 'SonarSource/sonarcloud-github-action@master' and sets environment variables for 'GITHUB\_TOKEN' and 'SONAR\_TOKEN'.

```
1 name: Build
2 on:
3   push:
4     branches:
5       - main
6   pull_request:
7     types: [opened, synchronize, reopened]
8 jobs:
9   sonarcloud:
10    name: SonarCloud
11    runs-on: ubuntu-latest
12    steps:
13      - uses: actions/checkout@v2
14        with:
15          fetch-depth: 0 # Shallow clones should be disabled for a better relevancy of analysis
16      - name: SonarCloud Scan
17        uses: SonarSource/sonarcloud-github-action@master
18        env:
19          GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN } # Needed to get PR information, if any
20          SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
```

# SONARCLOUD – PROJECT

20

```
SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
```

**Commit new file**

Create build.yml

Add an optional extended description...

☒ Commit directly to the `main` branch.

☐ Create a **new branch** for this commit and start a pull request. [Learn more about pull requests.](#)

**Commit new file** **Cancel**

# SONARCLOUD – PROJECT

The screenshot shows the SonarCloud interface for a project named 'prova\_1'. The left sidebar contains navigation links: 'Configure', 'Main Branch', 'Pull Requests', 'Branches', 'Information', 'Administration', and 'Collapse'. The main content area is titled 'Sabato Nocera > prova\_1 > Configure > With GitHub Actions'. It prompts the user to 'Create a sonar-project.properties file'. Below this, it instructs to 'Create a configuration file in the root directory of the project and name it sonar-project.properties'. A text box contains the following configuration:

```
sonar.projectKey=nocera-sabato_prova_1
sonar.organization=nocera-sabato

# This is the name and version displayed in the SonarCloud UI.
#sonar.projectName=prova_1
#sonar.projectVersion=1.0

# Path is relative to the sonar-project.properties file. Replace "\" by "/" on Windows.
#sonar.sources=.

# Encoding of the source code. Default is default system encoding
#sonar.sourceEncoding=UTF-8
```

A 'Copy' button is located to the right of the text box. Below the code, a flag icon is accompanied by the text 'And you are done!' and a message: 'You should see the page refresh itself in a few moments with your analysis results if everything runs successfully.' A green checkmark and the text 'You'll get a first analysis of your default branch' are at the bottom.

sonarcloud

prova\_1

PUBLIC

Configure

Main Branch

Pull Requests 0

Branches 0

Information

Administration

Collapse

My Projects My Issues Explore

Sabato Nocera > prova\_1 > Configure > With GitHub Actions

Create a sonar-project.properties file

Create a configuration file in the root directory of the project and name it sonar-project.properties

```
sonar.projectKey=nocera-sabato_prova_1
sonar.organization=nocera-sabato

# This is the name and version displayed in the SonarCloud UI.
#sonar.projectName=prova_1
#sonar.projectVersion=1.0

# Path is relative to the sonar-project.properties file. Replace "\" by "/" on Windows.
#sonar.sources=.

# Encoding of the source code. Default is default system encoding
#sonar.sourceEncoding=UTF-8
```

Copy

And you are done!

You should see the page refresh itself in a few moments with your analysis results if everything runs successfully.

✓ You'll get a first analysis of your default branch

# SONARCLOUD – PROJECT

The screenshot shows the GitHub interface for the repository 'nocera-sabato/prova\_1'. The repository is public and has 1 branch and 0 tags. The 'Add file' button is circled in green, and its dropdown menu is also circled in green, showing the options 'Create new file' and 'Upload files'. The repository contains two files: '.github/workflows' and 'SQLInjectionExample.java'. The right sidebar shows the 'About' section with no description, website, or topics provided, and 0 stars, 1 watching, and 0 forks. The 'Releases' section shows no releases published. The 'Packages' section shows no packages published. The 'Languages' section shows Java at 100.0%.

nocera-sabato / prova\_1 (Public)

Pin Unwatch 1 Fork 0 Star 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

Go to file Add file <> Code

Create new file Upload files

2 commits

nocera-sabato Create build.yml

.github/workflows Create build.yml 1 minute ago

SQLInjectionExample.java Create SQLInjectionExample.java 25 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README

About

No description, website, or topics provided.

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Languages

Java 100.0%

# SONARCLOUD – PROJECT

The screenshot shows the SonarCloud interface for a project named 'prova\_1' by user 'nocera-sabato'. The project is public. The 'Code' tab is selected, showing the 'sonar-project.properties' file. The file content is as follows:

```
1 sonar.projectKey=nocera-sabato_prova_1
2 sonar.organization=nocera-sabato
3
4 # This is the name and version displayed in the SonarCloud UI.
5 #sonar.projectName=prova_1
6 #sonar.projectVersion=1.0
7
8 # Path is relative to the sonar-project.properties file. Replace "\" by "/" on Windows.
9 #sonar.sources=.
10
11 # Encoding of the source code. Default is default system encoding
12 #sonar.sourceEncoding=UTF-8
13
```

The file editor interface includes a 'Cancel changes' button at the top right and a 'Preview changes' tab. The file content is displayed in a code editor with line numbers and syntax highlighting.

# SONARCLOUD – PROJECT

The screenshot shows the SonarCloud project page for 'prova\_1'. The left sidebar contains navigation links: Overview, Main Branch, Pull Requests (0), Branches (1), Information, Administration, and Collapse. The main content area displays project details for 'Sabato Nocera > prova\_1 > main'. The 'Summary' tab is active, showing 21 Lines of Code and a last analysis time of 2 minutes ago. The 'Quality Gate' status is 'Not computed'. Below this, several metrics are displayed in a grid:

- Reliability:** 2 Bugs (E)
- Maintainability:** 3 Code Smells (A)
- Security:** 1 Vulnerabilities (E)
- Security Review:** 2 Security Hotspots (E), 0.0% Reviewed
- Coverage:** 0.0% Coverage (E)
- Duplications:** 0.0% Duplications (E)

Two green circles highlight the 'Security' and 'Security Review' sections. The footer contains copyright information and links to Terms, Pricing, Privacy, Security, Community, Documentation, Contact us, Status, and About.

La pagina del progetto su SonarCloud viene aggiornata con le informazioni sulle issue ad ogni commit



# SONARCLOUD – PROJECT

The screenshot displays the SonarCloud web interface for a project named 'prova\_1'. The left sidebar contains navigation links for Overview, Main Branch, Pull Requests, Branches, Information, Administration, and Collapse. The main content area shows the 'Issues' tab for the 'main' branch. A filter is applied for 'Type' as 'Vulnerability'. The issue list shows one issue titled 'Revoke and change this password, as it is compromised.' which is highlighted with a green circle. The issue details show it is a 'Vulnerability' type, 'Open' status, with a severity of 'Minor' and an effort of '1h effort • 32 minutes ago'. The bottom of the page includes copyright information and links to Terms, Pricing, Privacy, Security, Community, Documentation, Contact us, Status, and About.

sonarcloud

prova\_1

PUBLIC

Overview

Main Branch

Pull Requests 0

Branches 1

Information

Administration

Collapse

My Projects My Issues Explore

Sabato Nocera > prova\_1 > main

Summary Issues Security Hotspots Measures Code Activity

Filters Clear All Filters

Type 1 x

- Bug 2
- Vulnerability 1
- Code Smell 3

Ctrl + click to add to selection

Severity

- Blocker 1
- Minor 0
- Critical 0
- Info 0
- Major 0

Resolution

Status

Bulk Change

1/1 issues 1h effort

SQLInjectionExample.java

Revoke and change this password, as it is compromised.

No tags

Vulnerability Open Blocker Not assigned

1h effort • 32 minutes ago

1 of 1 shown

© 2008-2023, SonarCloud by SonarSource SA. All rights reserved.

Terms Pricing Privacy Security Community Documentation Contact us Status About

# SONARCLOUD – PROJECT

The screenshot displays the SonarCloud web interface for a project named 'prova\_1'. The left sidebar contains navigation links: Overview, Main Branch, Pull Requests (0), Branches (1), Information, Administration, and Collapse. The main content area shows the 'Issues' tab for the file 'SQLInjectionExample.java'. A single issue is listed with the title 'Revoke and change this password, as it is compromised.' and a severity of 'Vulnerability'. The issue details panel on the right shows the title, a description stating 'Credentials should not be hard-coded', and a code snippet from the file. The code snippet is as follows:

```
6 // Carica il driver del database
7 Class.forName("com.mysql.jdbc.Driver");
8
9 // Connessione al database
10 Connection connection = DriverManager.getConnection(
11     "jdbc:mysql://localhost:3306/test_db", "root", "password"
12 );
13
14 // Query di esempio con input dell'utente
15 String userInput = "1 OR 1=1";
16 String query = "SELECT * FROM users WHERE id = " + userInput;
17
18 // Creazione della statement
19 Statement statement = connection.createStatement();
20
21 // Esecuzione della query
22 ResultSet result = statement.executeQuery(query);
23
24 // Stampa i risultati
```

The issue is highlighted with a green oval in the code snippet, and the text 'Revoke and change this password, as it is compromised.' is circled in red. The interface also shows a summary of the issue, including its severity, status, and effort.

# .github/workflows/build.yml

```
name: Build
on:
  push:
    branches:
      - main
  pull_request:
    types: [opened, synchronize, reopened]
jobs:
  sonarcloud:
    name: SonarCloud
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
        with:
          fetch-depth: 0 # Shallow clones should be disabled for a better relevancy of analysis
      - name: SonarCloud Scan
        uses: SonarSource/sonarcloud-github-action@master
        env:
          GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN } # Needed to get PR information, if any
          SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
```

# sonar-project.properties

```
sonar.projectKey=sabato-nocera_provaova
```

```
sonar.organization=sabato-nocera
```

```
# This is the name and version displayed in the SonarCloud UI.
```

```
#sonar.projectName=provaova
```

```
#sonar.projectVersion=1.0
```

```
# Path is relative to the sonar-project.properties file. Replace "\" by "/" on Windows.
```

```
sonar.sources=.
```

```
sonar.java.binaries=.
```

```
# Encoding of the source code. Default is default system encoding
```

```
#sonar.sourceEncoding=UTF-8
```