- Bonifica codice HTML.

- Rimuovere le interruzioni di linea, i caratteri di tabulazione (tab), gli spazi bianchi non necessari. Il "text/template" e "html/template" includono un modo per rimuovere gli spazi bianchi dal template, utilizzando un segno meno - all'interno del delimitatore dell'azione.
- URL Request Path. Nel pacchetto "net/http" c'è un tipo di multiplexer di richiesta HTTP chiamato ServeMux, che viene utilizzato per far corrispondere la richiesta in arrivo ai pattern registrati e quindi a invocare il gestore che più si avvicina all' URL richiesto. Oltre al suo scopo principale, si occupa anche di sanitizzare l'URL, reindirizzando qualsiasi richiesta contenente '.' o '..' o '/' ripetuti a un URL equivalente, ma più pulito. Di seguito un esempio di Mux:

```
func main() {

  mux := http.NewServeMux()
  rh := http.RedirectHandler("http://yourDomain.org", 307)

  mux.Handle("/login", rh)
  log.Println("Listening...")

  http.ListenAndServe(":3000", mux)

}
```

7.12.3.3 Gestione Sessione, Controlli Accessi e Crittografia

7.12.3.3.1 Sessioni

- La creazione della sessione deve essere eseguita su un sistema attendibile.
- Assicurarsi che gli algoritmi utilizzati per generare l'identificatore di sessione siano sufficientemente casuali al fine di prevenire una forzatura bruta di sessione.
- Una volta assicurato un token sufficientemente forte, impostare l'opportuno valore per i cookie: 'Domain', 'Path', 'Expires', 'HttpOnly' e 'Securè.
- Al momento del login, deve essere sempre generata una nuova sessione. La vecchia sessione non deve essere mai riutilizzata, anche se non è scaduta. Utilizzare anche il parametro "Expire" per eseguire la chiusura della sessione in modo da prevenire il "session hijacking". Un altro aspetto importante dei cookie è quello di impedire l'accesso simultaneo per lo stesso nome utente. Ciò può essere fatto mantenendo un elenco degli utenti connessi e confrontare il nuovo nome utente di accesso con tale elenco. Questo elenco di utenti attivi viene di solito persistito su un database.
- Gli identificatori di sessione non devono mai essere esposti negli URL. Questi dovrebbero essere localizzabili solo nei cookie presenti nell'intestazione http. Un esempio di cattiva pratica è quello di passare gli identificatori di sessione come parametri della GET. I dati della sessione devono inoltre essere protetti dall'accesso non autorizzato da parte di altri utenti del server.
- È necessario passare da HTTP a HTTPS, al fine di prevenire potenziali attacchi Man In The Middle (MITM), nei quali un attaccante si frappone fra due endpoint, "fiutando" i pacchetti in transito. In tal modo tutto il traffico è visibile e comprensibile, poiché l'HTTP prevede la trasmissione delle informazioni in chiaro. Utilizzando HTTPS in tutte le richieste (pacchetto "crypto/tls" di Go) la trasmissione risulta crittografata e il compito per l'attaccante molto più arduo.
- In caso di operazioni altamente sensibili o critiche, il token deve essere generato per richiesta invece che per sessione. Accertarsi sempre che il token sia sufficientemente casuale sia sufficientemente lungo da proteggerlo contro possibili attacchi di forza bruta.
- Aspetto da considerare nella gestione delle sessioni è la funzionalità Logout. L'applicazione deve fornire un modo per disconnettersi da tutte le pagine che richiedono l'autenticazione, nonché terminare completamente la sessione e le connessione ad esse associate. In particolare, quando un utente si disconnette, il cookie deve essere eliminato dal client. La stessa azione deve essere intrapresa dalla componente che si occupa della memorizzazione delle informazioni della sessione utente.

7.12.3.3.2 Controllo Accessi

- Utilizzare solo gli oggetti di sistema attendibili per le decisioni di autorizzazione all'accesso.
- Generare un token di sessione lato server, quindi memorizzare e utilizzare questo token per convalidare l'utente e applicare il modello predefinito di controllo degli accessi.
- Il componente utilizzato per l'autorizzazione di accesso deve essere un unico componente (centralizzazione), utilizzato a livello di sito. Ciò include quelle funzioni di libreria utilizzate che chiamano servizi di autorizzazione esterni.
- In caso di eccezione, il controllo degli accessi dovrebbe fallire in modo sicuro. A tale scopo è opportuno utilizzare la funzione 'Defer'.
- Se l'applicazione non può accedere alle informazioni di configurazione, ogni accesso all'applicazione deve essere negato.
- I controlli di autorizzazione devono essere applicati su ogni richiesta, inclusi gli script eseguiti lato server e le richieste provenienti da tecnologie lato client come AJAX o Flash.
- È importante separare correttamente la logica di gestione dei privilegi dal resto del codice applicativo.
- Altre operazioni importanti in cui i controlli di accesso devono essere attuati al fine di impedire ad un utente non autorizzato di accedervi, sono:
    - File e altre risorse,
    - Protezione URL’s,
    - Protezioni Funzioni,
    - Riferimenti diretti ad oggetti,
    - Servizi,
    - Dati applicativi,
    - Attributi utente e dati e informaizoni sulle policy.
- Se i dati di stato devono essere memorizzati lato client, è necessario utilizzare la crittografia ed effettuare opportuni controlli d'integrità per prevenire possibili manomissioni.
- Il flusso della logica applicativa deve essere conforme alle regole di business.
- Quando si trattano transazioni, il numero di transazioni che un singolo utente o dispositivo può eseguire in un dato periodo di tempo deve essere superiore ai requisiti previsti, ma sufficientemente basso da impedire all'utente di eseguire un attacco di tipo DoS.
- L'impiego della sola intestazione HTTP "referer" è insufficiente per convalidare l'autorizzazione e deve essere utilizzato solo come controllo supplementare.
- Per le sessioni con autenticazione a lungo termine, l'applicazione deve riesaminare periodicamente l'autorizzazione dell'utente per verificare che i permessi di quest'ultimo non siano cambiati. Se le autorizzazioni sono cambiate, è necessario scollegare l'utente e costringerlo a riautenticarsi.
- Gli account degli utenti devono essere verificati periodicamente, al fine di rispettare le procedure di sicurezza, (ad esempio, disabilitando l'account utente dopo 30 giorni dalla data di scadenza della password).
- L'applicazione deve supportare la possibilità di disabilitare gli account e la chiusura delle sessioni in caso di revoca dell'autorizzazione dell'utente, (ad es. cambiamento di ruolo, situazione occupazionale, ecc.).
- Gli account di servizio esterno, o che supportano connessioni da o verso sistemi esterni, devono essere dotati del più basso possibile livello di privilegi.

7.12.3.3.3 Crittografia e Hashing

La crittografia deve essere utilizzata ogni qual volta è necessario comunicare o memorizzare dati sensibili. Le regole da seguire sono le seguenti:

- Utilizzare algoritmi sicuri di hashing come l’SHA-256.
- Un caso di utilizzo "semplice" di crittografia è il protocollo HTTPS - Hyper Text Transfer Protocol Secure.
- AES è lo standard di fatto per quanto riguarda la crittografia a chiave simmetrica. Questo algoritmo, come molte altre cifrature simmetriche, può essere implementato in diverse modalità.