```
}
```

7.6.10.7 Definizione delle classi

Evitare l'utilizzo di classi Wrapper. Se il wrapper è degno di maggiore fiducia rispetto al codice che lo utilizza, si può aprire un insieme unico di debolezze di sicurezza. Qualsiasi cosa fatta per conto di un chiamante, le cui autorizzazioni limitate non sono incluse nel controllo di sicurezza appropriato, è una potenziale debolezza da sfruttare.
Mai abilitare qualcosa attraverso il Wrapper che il chiamante non potrebbe fare su se stesso. Questo è un pericolo quando si effettua qualcosa che comporta un controllo di sicurezza limitato. Quando i controlli a livello singolo sono coinvolti, l'interposizione del codice di Wrapper tra il chiamante reale e l'elemento API in questione può facilmente causare il controllo di sicurezza per avere successo se non dovrebbe, quindi indebolire la sicurezza.

7.6.10.8 User input

I dati utente, qualsiasi tipo di input (dati da una richiesta Web o un URL, input ai controlli di un'applicazione Microsoft Windows Form e così via), possono influenzare negativamente il codice perché spesso vengono utilizzati direttamente come parametri per chiamare altro codice. Questa situazione è analoga a un modulo dannoso che chiama il codice con parametri estranei, e dovrebbero essere prese le stesse precauzioni.
Per cercare questi possibili bug, cercate di immaginare quali siano i valori possibili e valutare se il codice che visualizza questi dati possa gestire tutti questi casi. È possibile risolvere questi bug attraverso 'adozione della tecnica della white list, per cui si accettano solo i dati previsti e si rifiutano tutti gli altri.

Alcune considerazioni importanti che coinvolgono i dati utente includono quanto segue:

- tutti i dati contenuti in una risposta del server vengono eseguiti sulla pagina web dal client. Se il tuo web server prende i dati utente e li inserisce nella pagina Web restituita, potrebbe includere ad esempio un tag <script> e ed essere eseguito (attacco XSS);
- il client può richiedere qualsiasi URL;
- la funzione eval(datiUtente) può fare qualsiasi cosa;
- fare attenzione ai nomi utente che potrebbero avere più di un formato canonico. Ad esempio, in Microsoft Windows 2000, è possibile utilizzare spesso il modulo di nome utente MYDOMAIN \ o il modulo username@mydomain.example.com;
- prendere in considerazione i percorsi ingannevoli o non validi: quelli forniti di ripetuti "..\" e quelli molto lunghi;
- può esserci un uso sconsiderato del carattere (*);
- fare attenzione all'espansione dei token (% token%);
- verificare che non vi siano strani forme di percorsi con un significato speciale;
- appurare la correttezza delle versioni brevi di nomi file lunghi, come longfi ~ 1 per longfilename.

7.6.10.9 Concorrenza

Utilizzo di synchronized per la gestione della memoria

Se un metodo di una classe `Dispose` non è `synchronized`, è possibile che il codice di cleanup all'interno di Dispose sia eseguito più di una volta, come illustrato nell'esempio seguente.
Esempio:

```
void Dispose()
{
  if( myObj != null )
  {
    Cleanup(myObj);
    myObj = null;
  }
}
```