]HackingTeam[ | gesi RAS

**GE.S.I.**

# Vulnerability Assessment

# Poms

**Milano, 19 Luglio 2007**

# Indice

# Descrizione dell'attività

L'attività consiste nell'analisi della sicurezza dell'applicativo Poms, utilizzato dall'interno della rete informatica del cliente per svolgere operazioni di visualizzazione riguardanti contenuti finanziari.

L'applicativo si compone principalmente di una maschera di ricerca attraverso la quale è possibile restringere il numero di informazioni visualizzate in base a determinati criteri scelti dall'utente.
L'autenticazione è fornita da un sistema centralizzato esterno all'applicativo in esame, e non esistono zone dove sia possibile compiere operazioni dispositive.

L'analisi è stata svolta in ambiente di produzione in quanto unica versione completamente aggiornata e comprensiva dei dati corretti, utilizzando come partenza l'indirizzo *http://areabanca.servizi.ras/Poms/*.

# Analisi con metodologia Black-Box

## *Test complessivi*

L'applicativo non presenta nessuna sezione che sia fruibile senza prima autenticarsi al sistema.

Tutti gli attacchi effettuati al fine di accedere a parti dell'applicativo senza fornire credenziali valide di accesso hanno avuto esito negativo.

## *Forzatura della login*

Il sistema di login è fornito da un sistema centralizzato esterno all'applicativo, e non risulta comunque vulnerabile ad attacchi volti a eludere l'autenticazione.

## *Riepilogo vulnerabilità/soluzioni*

Nessuna vulnerabilità riscontrata.

# Analisi con metodologia White-Box

Le credenziali di accesso per l'analisi dell'applicativo sono state fornite dal cliente, e corrispondono ad un utente con livelli normali di accesso alle informazioni.

| | |
|---|---|
| **Username** | eutz161 |
| **Password** | PRFstrng |

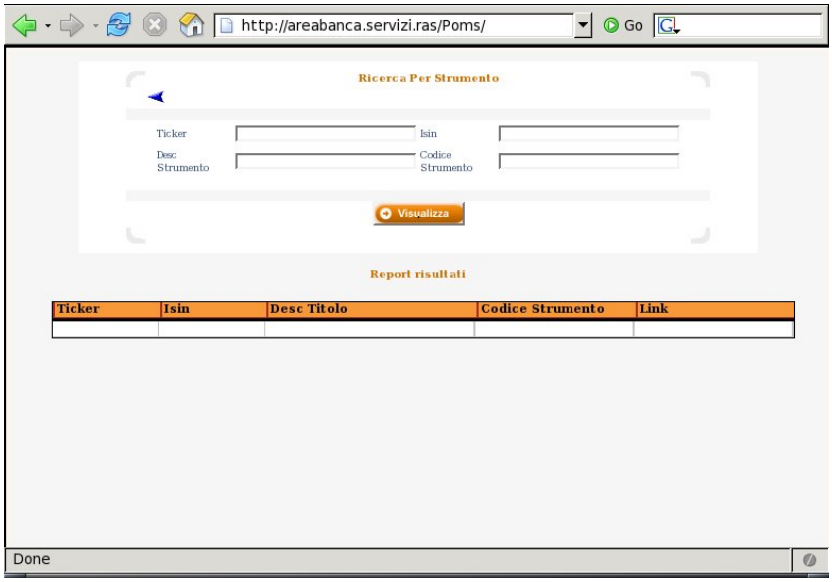
## *Test complessivi*

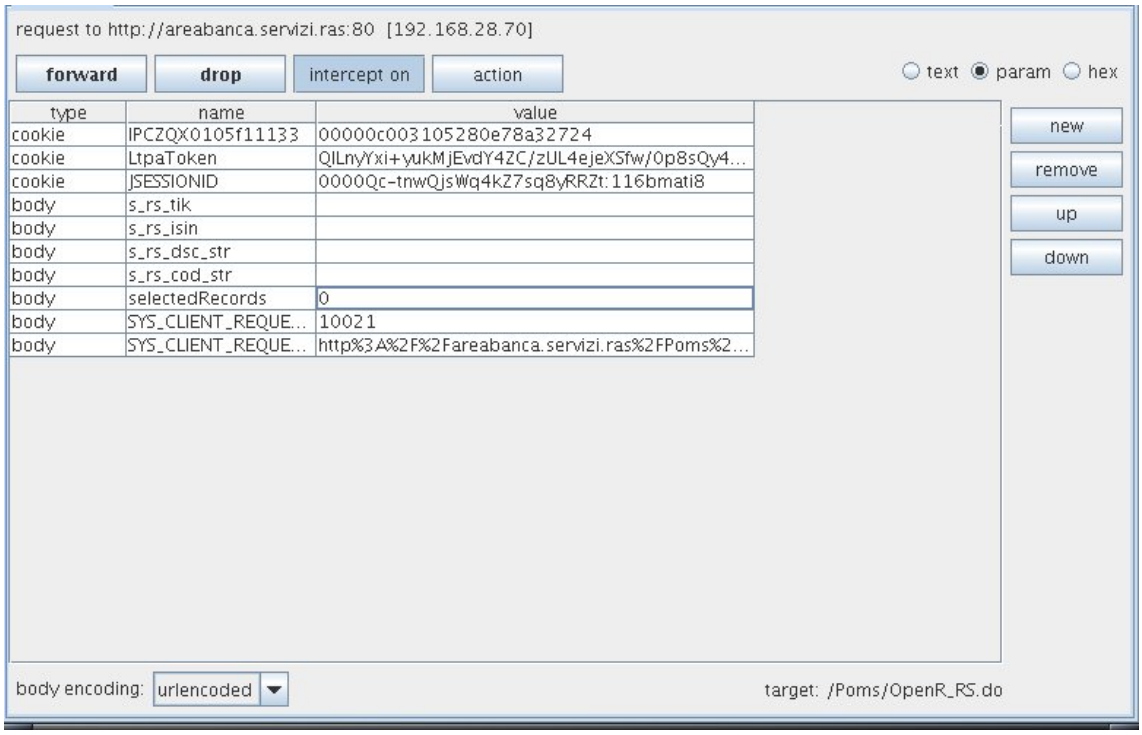
L'applicativo risulta vulnerabile ad un attacco di tipo *Cross Site Scripting* (XSS) effettuabile sulla pagina secondaria di ricerca.

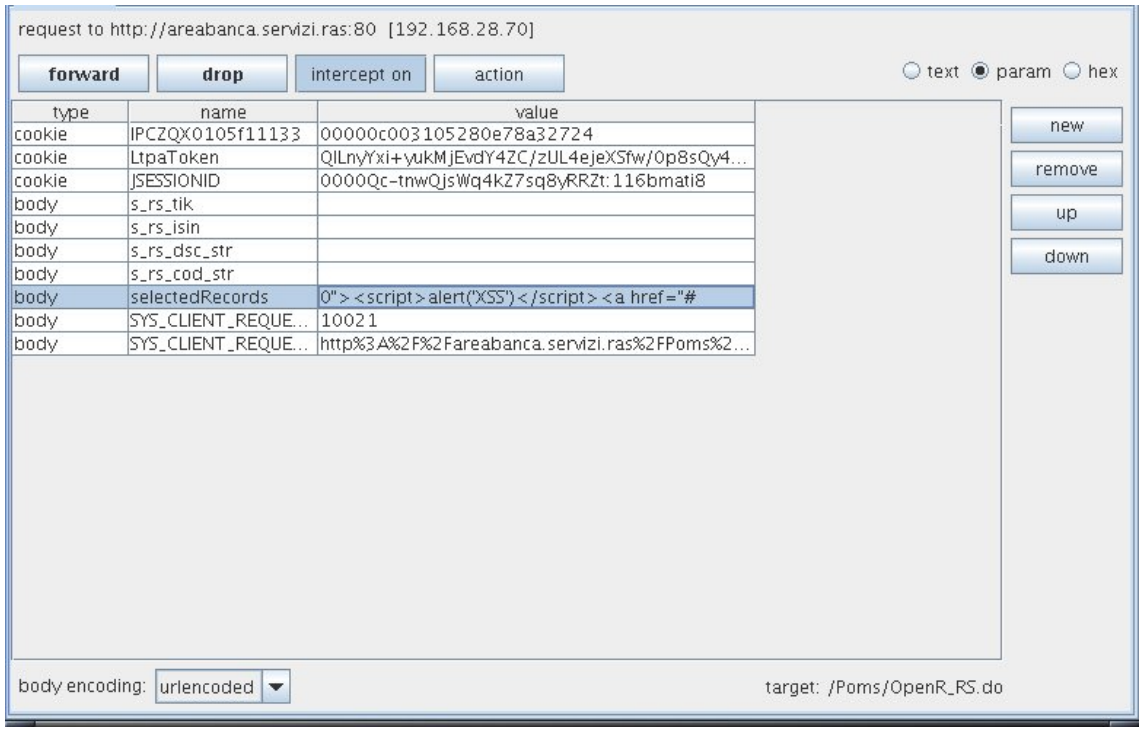
L'attacco è di tipo non persistente, quindi per poterlo effettuare è necessario per un attaccante trovare il modo di far eseguire alla vittima una richiesta opportunamente modificata (tramite ad esempio un collegamente presente in un'email o in un altro sito).

Il problema consiste nella mancanza di validazione del parametro **selectedRecords** da parte delle pagine **OpenR.do** e **OpenR_RS.do**, dove è possibile aggiungere codice arbitrario che sarà inviato al client come risposta alla richiesta, consentendone l'esecuzione sul sistema vittima.

Dall'analisi svolta presso il cliente, è risultato che il problema è relativo ad un controllo dell'architettura utilizzata per costruire l'applicativo in esame, e che quindi la correzione deve essere effettuata direttamente sul controllo stesso in modo da eliminare tutte le vulnerabilità relative.






## *Sicurezza delle sessioni*

Le sessioni si basano su un *cookie* di sessione rilasciato al client durante la fase di autenticazione ed utilizzato successivamente per garantire l'accesso alle risorse consentite.

Il sistema risulta sicuro, sebbene non esista una procedura esplicita di *logout*.
Questo viene infatti effettuato in maniera automatica al momento dell'uscita dall'applicativo tramite chiusura del browser o inserimento di un nuovo indirizzo nel browser.
La procedura funziona però correttamente solo se viene utilizzato il browser Microsoft Internet Explorer, mentre con altri browser diffusi (ad esempio Mozilla Firefox) non viene invece invocata correttamente, e quindi è possibile utilizzare il *cookie* di sessione precedentemente rilasciato per ottenere un accesso successivo senza necessità di effettuare nuovamente l'autenticazione al sistema.

## *Riepilogo vulnerabilità/soluzioni*

### *Cross Site Scripting* (XSS)

**Problema**

Mancata validazione dei parametri in ingresso.

**Descrizione**

Le pagine **OpenR.do** e **OpenR_RS.do** non controllano la correttezza del contenuto del campo **selectedRecords** prima di visualizzarlo all'interno del contesto corrente.

**Impatto**

Possibilità di eseguire codice arbitrario sul sistema vittima e di modificare la pagina risultante.

**Soluzione**

Effettuare il controllo del parametro come viene fatto per gli altri parametri della pagina.

## Mancanza di una procedura di *logout* esplicita

**Problema**

Mancanza di una funzionalità esplicita per terminare la sessione da parte dell'utente.

**Descrizione**

Non è presente una funzionalità per effettuare il *logout* in maniera esplicita, con un pulsante apposito o un collegamento adatto.

La procedura viene richiamata correttamente solo nel caso dell'abbandono dell'applicazione nel caso si utilizzi il browser Microsoft Internet Explorer.

**Impatto**

Se non viene richiamata la procedura di *logout*, la sessione rimane attiva sul server e potrebbe quindi permettere ad un attaccante in grado di ottenere il *cookie* di sessione di impersonare l'utente ed effettuare le operazioni con i suoi privilegi di accesso.

**Soluzione**

Introdurre una procedura esplicita per terminare la sessione dell'utente oppure rendere compatibile la procedura presente con tutti i browser che potrebbero essere usati sull'applicativo.

## Considerazioni conclusive

Il livello dell'applicativo è risultato adeguato, anche se sono state riscontrate alcune lacune isolate che potrebbero comprometterne la sicurezza.
Una volta risolti questi problemi, la struttura si presenterà robusta e non vulnerabile ad attacchi.

## Sintesi finale sugli esiti del test ed eventuali raccomandazioni

Si raccomanda di correggere il problema della mancanza di una procedura esplicita di*logout* introducendo una funzionalità apposita all'interno dell'applicativo sottoforma di pulsante o di collegamento, anche se il *logout* effettuato in maniera automatica all'uscita dall'applicativo è una caratteristica utile che può essere mantenuta, magari estendendola per renderla compatibile con un numero maggiore di browser che potrebbero essere supportati in futuro.

# Appendice: terminologia utilizzata

Gli attacchi di livello applicativo sfruttano vulnerabilità (sia di natura logico-architetturale, sia di natura implementativa) per indurre nelle applicazioni comportamenti anomali, le cui conseguenze possono essere le più disparate: crash dell'applicazione, furto di dati, accesso ai sistemi su cui le applicazioni sono eseguite, ecc.

Le tecniche di attacco che sono state utilizzate sono descritte nei seguenti sottoparagrafi.

## *Cross site scripting*

- **Obiettivo**: furto d'identità ai danni di utenti di applicazioni web che fanno uso di cookie per la gestione delle sessioni.
- **Attaccanti**: chiunque sia interessato al furto dell'identità di un utente autorizzato (che abbia stabilito una sessione con l'applicazione web).
- **Descrizione**: si tratta di una tecnica che, mediante l'inserimento di elementi di scripting nei parametri inviati all'applicazione, provoca l'esecuzione degli stessi da parte del browser della vittima. Gli elementi di scripting causano l'invio dei cookie settati dall'applicazione target sul browser della vittima verso server un HTTP sotto il controllo dell'attaccante. Solitamente l'obiettivo dell'attacco è il cookie di sessione della vittima. La conoscenza di questo cookie permette infatti di sottoporre richieste all'applicazione utilizzando l'identità della vittima. Gli attacchi di cross site scripting sono possibili quando l'applicazione web restituisce al browser (per normale logica di funzionamento o a causa di una condizione di errore) parametri sottoposti dall'utente in una precedente richiesta.
- **Condizioni necessarie per l'attacco**: assenza di controlli sull'input ricevuto ed errori relativi all'escaping di metacaratteri nell'HTML ritornato al browser.
- **Probabilità di successo**: questo attacco richiede l'uso di tecniche di social engineering per indurre la vittima a stabilire una sessione con l'applicazione target e sottoporre ad essa una richiesta contenente il codice malizioso. Frequentemente questo viene fatto per mezzo di email che invitano a seguire un link verso l'applicazione target. La probabilità di successo di tali attacchi è solitamente bassa.
- **Aspetti facilitanti**: in alcuni casi è possibile inserire elementi di scripting in parametri che vengono salvati su database e restituiti all'utente ad ogni successiva richiesta (database XSS). Questo determina l'invio di cookie verso il server HTTP sotto il controllo dell'attaccante ogni volta che la vittima accede all'applicazione ed aumenta in modo considerevole la probabilità che l'attaccante riesca ad utilizzarlo con successo.

- **Contromisure**: gli attacchi di tipo XSS possono essere neutralizzati mediante le seguenti tecniche:
  - filtraggio dei parametri in input, mediante filtri che eliminano dall'input i metacaratteri utilizzati in HTML e linguaggi di scripting client-side (<, >, apici, ecc.);
  - escaping dei metacaratteri contenuti nei parametri in input che devono essere inseriti in pagine HTML restituite al browser degli utenti.

## *SQL Injection*

- **Obiettivo**: gli attacchi basati su SQL injection possono avere diversi obiettivi:
  - accesso ad informazioni riservate memorizzate sui database server che costituiscono il data layer dell'architettura applicativa attaccata;
  - accesso non autorizzato all'applicazione, aggirando il meccanismo di autenticazione;
  - esecuzione di comandi sui server del data layer.
- **Attaccanti**: utenti autorizzati che mirano ad accedere ad informazioni per le quali non possiedono diritti di accesso; utenti non autorizzati.
- **Descrizione**: si tratta di tecniche di manipolazione dei parametri in input utilizzati dall'applicazione per eseguire query SQL sul database. Lo scopo è sovvertire la logica della query in modo da ottenere:
  - messaggi di errore contenenti informazioni sulla struttura del database utilizzato;
  - informazioni differenti da quelle che la query dovrebbe estrarre;
  - recordset vuoti o tali da produrre un malfunzionamento dei meccanismi di autenticazione, allo scopo di accedere all'applicazione senza essere in possesso di credenziali valide.
- **Condizioni necessarie per l'attacco**: mancanza di filtri di validazione dell'input, che eliminano dai parametri inviati dall'utente token pericolosi, come parole chiave riservate del linguaggio SQL (ad esempio, SELECT, OR, ecc.).
- **Probabilità di successo**: fortemente dipendenti dalla logica applicativa.
- **Aspetti facilitanti**: la visualizzazione, sul lato client, dei messaggi di errore relativi all'accesso al database permette all'attaccante di raccogliere informazioni sulla sua struttura, aumentando significativamente le probabilità di successo.
- **Contromisure**: gli attacchi di tipo SQL injection possono essere neutralizzati mediante le seguenti tecniche:
  - filtraggio dei parametri in input, mediante filtri che eliminano dall'input token riservati e metacaratteri del linguaggio SQL;

- gestione degli errori di accesso al layer di accesso ai dati, allo scopo di intercettare e bloccare la visualizzazione lato client dei messaggi di errore.

## *Path traversal*

- **Obiettivo**: browsing di directory presenti sul web server ma non appartenenti alle applicazioni web pubblicate su di esso, per le quali non è previsto l'accesso da parte degli utenti.
- **Attaccanti**: questo tipo di attacco può essere portato da chiunque possa stabilire una connessione HTTP verso i server su cui è pubblicata l'applicazione.
- **Descrizione**: un attacco di path traversal consiste nella sottomissione di richieste verso il web server per risorse il cui URL contiene path non appartenenti alle applicazioni web pubblicate su di esso. Poiché in generale tali path non sono noti all'attaccante, essi vengono specificati in forma relativa, partendo dalla posizione di risorse note ed utilizzando sintassi del tipo "../../" per navigare a ritroso il file system. Si noti che questo attacco non è in alcun modo correlato alla logica applicativa, ma sfrutta eventuali vulnerabilità del server HTTP.
- **Condizioni necessarie**: queste tecniche possono essere utilizzate in presenza di web server sui quali non sono installate le security patch opportune; in ogni caso, la loro applicabilità non dipende dalla particolare applicazione pubblicata sul web server.
- **Probabilità di successo**: dipende dall'accuratezza della manutenzione del web server.
- **Aspetti facilitanti**: nessuno
- **Contromisure**: aggiornamento dei web server mediante applicazione delle opportune patches.

## *Cookie poisoning*

- **Obiettivo**: gli obiettivi possono essere molteplici; essi dipendono dalla logica dell'applicazione attaccata. In generale, le tecniche di cookie poisoning mirano a provocare comportamenti non previsti nell'applicazione attaccata in modo da poter interagire con essa in modi non previsti dal programmatore.
- **Attaccanti**: gli attacchi di cookie poisoning possono provenire da qualsiasi utente sul cui browser l'applicazione setta cookie.
- **Descrizione**: le tecniche di cookie poisoning consistono nella modifica dei dati contenuti nei cookie inviati dall'applicazione all'utente, allo scopo di produrre errori e/o portare l'applicazione in stati non correttamente gestiti quando i cookie sono restituiti al server. Per essere in grado di apportare le modifiche opportune, l'attaccante deve conoscere la logica con cui i dati contenuti nei cookie sono processati dall'applicazione.

- **Probabilità di successo**: dipendenti dal livello di conoscenza da parte dell'attaccante della logica di elaborazione dei dati contenuti nei cookie.
- **Aspetti facilitanti**: la memorizzazione nei cookie di parametri critici dal punto di vista della sicurezza è un errore comune, che rende pericolosi gli attacchi di cookie poisoning.
- **Contromisure**: limitare l'uso dei cookie alla memorizzazione di informazioni non critiche; nel caso questo non sia possibile, devono essere utilizzate tecniche (ad esempio crittografia) per impedire la modifica dei cookie.

## *Forceful browsing*

- **Obiettivo**: accesso non autorizzato a pagine e/o funzionalità dell'applicazione.
- **Attaccanti**: chiunque non sia in possesso di credenziali per accedere a tali pagine/funzionalità.
- **Descrizione**: gli attacchi di forceful browsing consistono semplicemente nella sottomissione di richieste HTTP per URL corrispondenti a pagine protette, senza seguire il percorso di navigazione previsto dal programmatore e, in particolare, aggirando le pagine di autenticazione.
- **Probabilità di successo**: dipendenti dal livello di conoscenza da parte dell'attaccante della struttura dell'applicativo. Tale livello può essere molto elevato per ex utenti che sono stati disabilitati.
- **Aspetti facilitanti**: messaggi di errore non propriamente gestiti dall'applicazione possono contenere informazioni sulla struttura delle directory dell'applicazione sul web server e semplificare la costruzione degli URL da utilizzare per compiere l'attacco.
- **Contromisure**: implementare una logica di controllo dello stato della sessione che impedisca l'accesso ad ogni parte dell'applicazione ad utenti non associati a sessioni autenticate.