]HackingTeam[| gesi RAS

GE.S.I.

Vulnerability Assessment

Full Casa

Milano, 22 Dicembre 2006

Indice

Descrizione dell'attività

Il preventivatore Full Casa permette di calcolare il proprio preventivo per l'assicurazione sulla casa tramite un portale disponibile via Internet.
Oltre alle funzionalità di calcolo dei valori economici in base alle scelte dell'utente, è possibile salvare il preventivo in maniera persistente presso le infrastrutture del cliente e ricevere una copia del preventivo tramite posta elettronica.

L'attività prevede l'analisi della sicurezza applicativa del portale che offre le funzionalità descritte precendentemente, al fine di identificare le vulnerabilità che potrebbero permettere ad un attaccante di alterare le informazioni riguardo al proprio preventivo, oppure di entrare in possesso di informazioni appartenenti ad altri utenti.

L'attacco viene effettuato esclusivamente in modalità Black-Box, in quanto non è presente nessun'area accessibile tramite autenticazione dell'utente.

L'attacco è assimilabile a quello effettuato nelle seguenti situazioni:

- analisi con metodologia black-box: nessun privilegio particolare, attaccante generico che accede tramite Internet all'applicativo

L'analisi è stata svolta utilizzando sia strumenti automatici di scansione di vulnerabilità, sia agendo manualmente per approfondire gli aspetti rilevati e per testare la presenza di vulnerabilità non standard all'interno dell'applicativo.

Le conoscenze pregresse, fornite dal Cliente, sono le seguenti:

- URI dell'applicativo (http://www.stage.ras/preventivi/casa/test)

L'analisi è stata svolta in un ambiente di preproduzione presso le infrastrutture del Cliente.

Analisi con metodologia Black-Box

L'attività consiste principalmente nel controllo di vulnerabilità presenti all'interno delle pagine dell'applicativo, specialmente nei form dove l'utente può fornire valori arbitrari, suddivisa in due fasi:

- controllo esteso a tutti i form presenti tramite scanner automatizzati per la ricerca di vulnerabilità comuni
- controllo manuale dei form sensibili che possono presentare vulnerabilità e di tutte le problematiche eventualmente rilevate durante la fase di scansione automatica

Questo tipo di attività permette quindi di individuare le vulnerabilità presenti, senza il rischio di tralasciare qualche form: la fase del processo automatizzato infatti prevede che, partendo dalla pagina iniziale, si crei un albero delle pagine presenti nell'applicazione, effettuando una ricerca esaustiva di tutti i collegamenti presenti nelle pagine analizzate.
Inoltre, vengono effettuati vari tentativi per testare vulnerabilità non identificabili in maniera ordinaria, cercando di trovare pagine con nomi standard anche se queste non sono collegate a nessuna pagina dell'applicativo, al fine di identificare possibili accessi amministrativi nascosti, o copie di backup dei file che potrebbero fornire indicazioni utili per l'attacco.

L'intervento umano viene poi impiegato per verificare se le vulnerabilità identificate esistano realmente o se siano solo dei falsi positivi, e per applicare un processo di ricerca dipendente dalla logica e dall'intuito, condizione spesso molto vicina a quella utilizzata da un reale attaccante durante il suo tentativo di intrusione.

Test complessivi

L'analisi ha evidenziato un problema di sicurezza relativo ad uno dei form utilizzati all'interno dell'applicativo.
Sono infatti possibili attacchi di tipo Cross site scripting (XSS) all'interno della ricerca del CAP associato ad un determinato Comune.
Utilizzando opportuni URI, un attaccante può essere in grado di alterare il normale comportamento della navigazione di un utente e compiere operazioni illegali sfruttando la vulnerabilità.
Questo tipo di attacco non prevede comunque la permanenza del codice dell'attaccante sull'infrastruttura del portale, e quindi può essere effettuato solamente verso utenti che vengono portati a visitare il portale tramite URI appositamente creati dall'attaccante (phishing).

Riepilogo vulnerabilità/soluzioni

Cross site scripting nella form di ricerca CAP

Problema

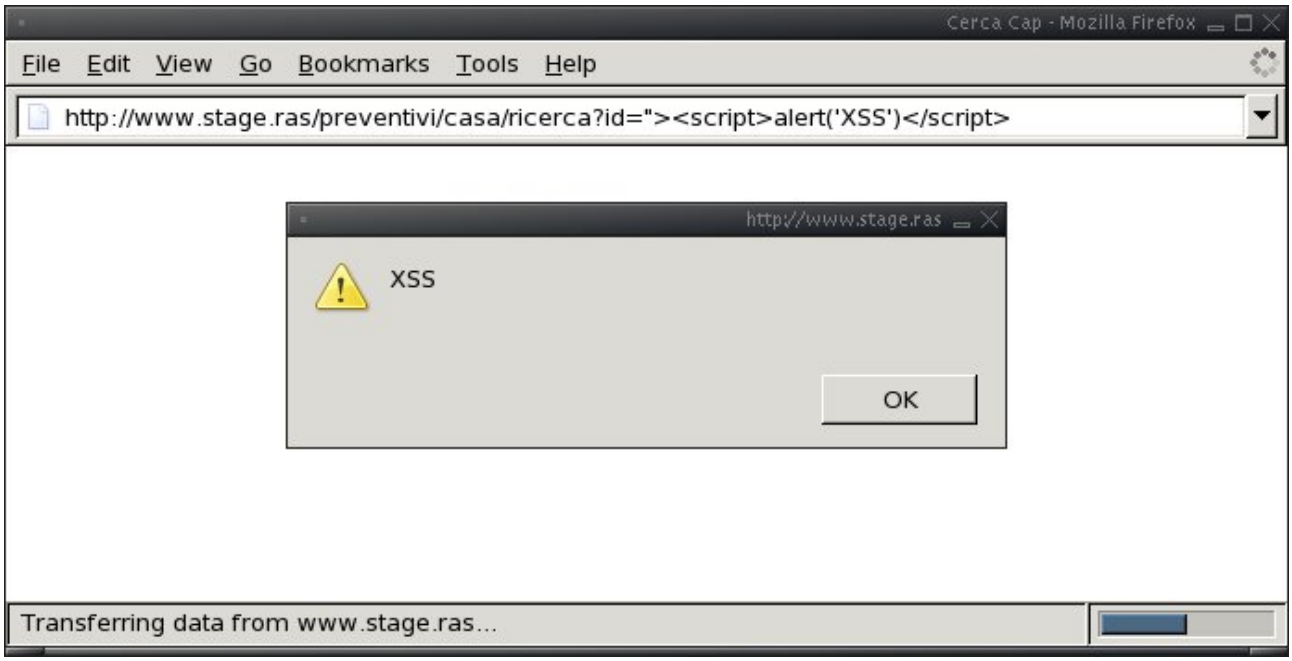
Il form si presenta vulnerabile ad attacchi di tipo Cross site scripting su due parametri utilizzati.

Descrizione

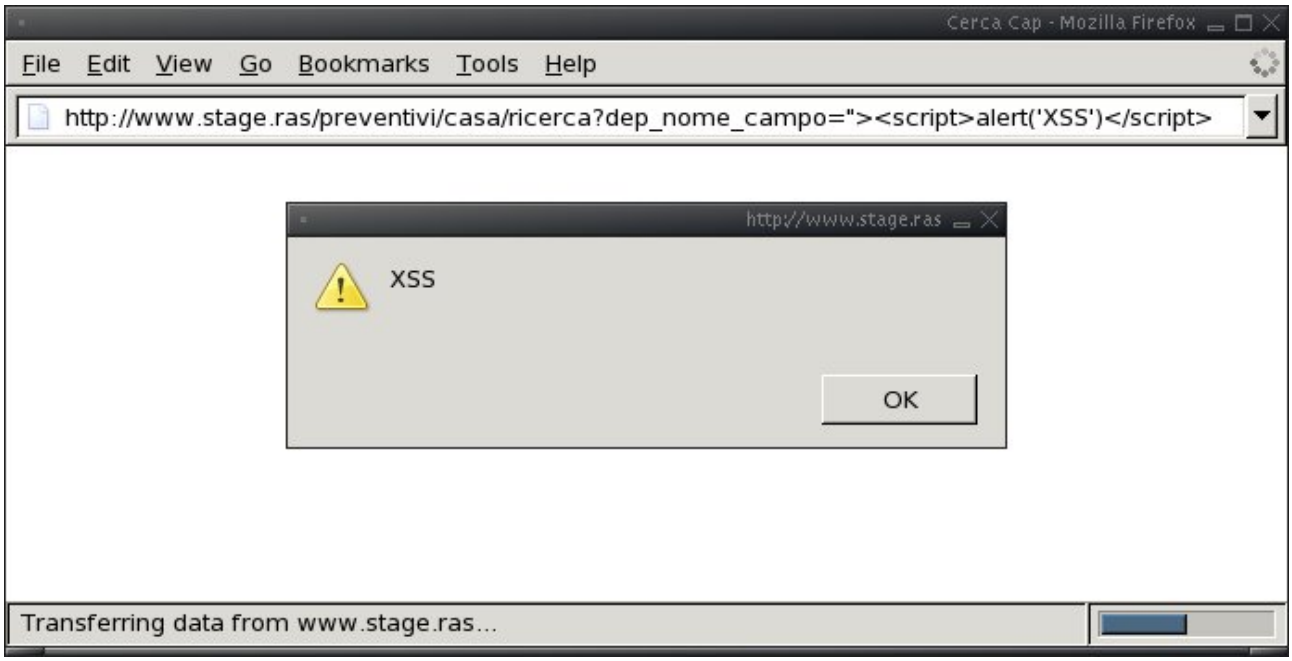
Il form vulnerabile è reperibile all'URI http://www.stage.ras/preventivi/casa/ricerca, ed i parametri vulnerabili sono i seguenti:

- id
- dep_nome_campo

Esempi di attacco possono essere i seguenti URI:



- http://www.stage.ras/preventivi/casa/ricerca?id=%22%3E%3Cscript%3Ealert('XSS')%3C/script%3E



- http://www.stage.ras/preventivi/casa/ricerca?dep_nome_campo=%22%3E%3Cscript%3Ealert('XSS')%3C/script%3E

Impatto

La vulnerabilità permette ad un attaccante di ottenere informazioni personali riguardo alla vittima una volta che questa è stata portata a seguire un deteminato collegamento al sito del portale vulnerabile, permettendo di appropriarsi di informazioni riservate contenute nel sistema della vittima, come credenziali di accesso o cookie di sessione che potrebbero permettere all'attaccante di prendere il controllo anche di altre applicazioni disponibili sullo stesso sito.

Soluzione

La soluzione consiste nel controllare i valori del campi inviati dall'utente prima che questi vengano processati, in maniera da evitare che contengano caratteri speciali e codici HTML.

La miglior soluzione è definire quale sia il formato esatto dei valori ammissibili per quegli specifici campi, e controllare in questo modo la validità dei dati inviati dall'utente.

Considerazioni conclusive

Il portale ha dimostrato di essere sicuro, ad eccezione della vulnerabilità esposta precendentemente.
Non sono quindi presenti vulnerabilità sfruttabili per compiere operazioni illecite.

Sintesi finale sugli esiti del test ed eventuali raccomandazioni

La raccomandazione principale a seguito di questa analisi è di impostare sempre a livello applicativo dei controlli per validare la correttezza e la sicurezza dei dati immessi dall'utente, utilizzando un metodo che permetta una facile estensione del meccanismo a nuovi campi ed a nuovi form che potranno essere aggiungi in seguito ad un ulteriore sviluppo dell'applicativo.

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.