]HackingTeam[

Milano, 21 Aprile 2009

Spett.le
Gucci Logistica spa
Via Don Lorenzo Perosi, 6
50018 Scandicci (FI)

Offerta n. 20090303.027-3.AL

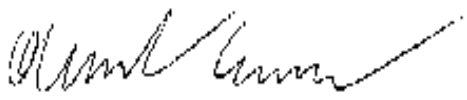
Alla cortese attenzione: Sigg. Andrea Pertici, Leonardo Rosa

Oggetto: Offerta Ethical Hacking

A seguito della Vs. gentile richiesta e degli approfondimenti intercorsi, Vi sottoponiamo la nostra proposta per quanto in oggetto
In attesa di un Vostro gradito riscontro, Vi porgiamo i nostri più cordiali saluti.

HT S.r.l.

Alessandro Lomonaco
Key Account Manager



Offerta Ethical Hacking

Sommario

1. RICHIESTA DEL CLIENTE

Il Cliente richiede di formulare una proposta relativa ad intervento di Ethical Hacking sulla propria infrastruttura.

In altre parole, si richiede una consulenza di Security Assessment che verifichi, secondo una logica indipendente e supra-partes, l'*effettiva* sicurezza di due applicativi Web e del perimetro interno.

2. SOLUZIONE PROPOSTA

L'intervento proposto comprende un attività di Ethical Hacking relativo a:

2.1. Penetration test Applicativo

Elenchiamo per praticità gli applicativi oggetto di verifica:

Applicativi	URL principale	Alias
CIM		
Check in Check Out		

Approccio: l'attività si svolgerà sia in modalità black box (senza credenziali) sia in modalità grey box.
Vincoli: non deve essere effettuata alcuna attività di verifica DoS, né di verifica buffer overflow.
Effettuazione test: nei laboratori HT, in giorni feriali, durante gli orari lavorativi.
Condizione: IPS/IDS disabilitati.

2.2. Penetration test Interno

Oggetto della verifica saranno 1 server Linux e 1 Server Win. Verrà inoltre verificato lo stato di sicurezza di una postazione tipo di un dipendente.

Approccio: l'attività si svolgerà sia in modalità black box (senza credenziali) per i 2 server, white box per la postazione tipo
Vincoli: non deve essere effettuata alcuna attività di verifica DoS, né di verifica buffer overflow.
Effettuazione test: presso la sede Gucci di Scandicci, in giorni feriali, durante gli orari lavorativi.
Condizione: IPS/IDS disabilitati.

3. METODOLOGIA DELLA SOLUZIONE PROPOSTA

3.1. Security Probe

Un attacco compiuto da hacker reali segue di norma la traccia che segue. Le attività di Ethical Hacking da noi eseguite tentano di emulare al 100% il comportamento di un vero hacker. Di seguito sono riportate le metodologie rispettivamente per la verifica network dall'esterno, per la verifica applicativa. Esse contemplano un livello di approfondimento notevole.

3.1.1. Analisi non invasiva

3.1.1.1. Footprinting

Questa fase ha lo scopo di raccogliere il maggior numero di informazioni sull'obiettivo che si intende attaccare senza "toccare" l'obiettivo stesso, ovvero effettuando una cosiddetta "analisi non invasiva". In particolare in questa fase si cerca di determinare: *domini, blocchi di rete e indirizzi IP dei sistemi direttamente collegati a Internet*. Gli strumenti utilizzati sono: Search Engine, Whois server, Arin database, interrogazione DNS, ecc.

3.1.1.2. Scanning

L'obiettivo dello scanning è ottenere una mappa più dettagliata possibile del sistema da attaccare. Ciò significa acquisire informazioni su quali indirizzi IP dei blocchi di rete trovati nella fase precedente siano effettivamente "contattabili" dall'esterno (IP discovery), quali servizi siano "attivi" (TCP/UDP port scan) e, infine, quali sistemi operativi "posseggano". Gli strumenti utilizzati sono: interrogazioni ICMP (gping, fping, ecc.), scansione delle porte TCP e UDP (strobe, netcat, nmap, ecc.), fingerprint dello stack (nmap, ethercap).

3.1.2. Analisi invasiva

3.1.2.1. Enumeration

Con questa fase si inizia l'"analisi invasiva". Si effettuano, infatti, connessioni dirette ai server e "interrogazioni" esplicite. Tali attività potrebbero, a seconda della configurazione presente sui sistemi target, originare dei logs sui sistemi (tipicamente su sistemi di controllo). Attraverso l'enumerazione si vuole giungere a identificare, sulle

macchine riscontrate come raggiungibili, account validi (list user accounts), risorse condivise (list file shares) e applicazioni attive sulle porte in ascolto (identify application). Le tecniche utilizzate variano a seconda dei sistemi operativi delle macchine che vogliamo analizzare.

3.1.3. Attacco

3.1.3.1. Gaining access

Una volta ottenute le informazioni del punto precedente, inizia il vero e proprio attacco che ha come obiettivo riuscire a "entrare" nel sistema remoto. I metodi utilizzati anche in questo caso dipendono dal sistema operativo della macchina target, ma si basano sostanzialmente sulla ricerca di password corrispondenti agli utenti trovati (password guessing), sullo sfruttamento di errori progettuali delle applicazioni e servizi attivi sul server (buffer overflows, attacchi data driven, ecc.) o del sistema operativo stesso.

3.1.3.2. Escalating privileges 1

L'obiettivo di questa fase è sfruttare i risultati ottenuti nella fase precedente per ottenere il pieno controllo del sistema remoto attaccato. Ciò si ottiene, per esempio, reperendo i files presenti sul sistema che contengono le password (/etc/passwd, SAM, ecc.) e tentando di decifrare le password in essi contenute (password cracking), oppure utilizzando appositi exploits.

3.1.4. Consolidamento

3.1.4.1. Pilfering

Se si giunge a questa fase significa che si è ottenuto il pieno controllo del sistema target. Quindi è bene valutare la configurazione del sistema stesso al fine di capire se, dove e cosa il sistema registra (logs). I sistemi di auditing saranno eventualmente disabilitati (es. con Win NT mediante auditpol). A questo punto la macchina in oggetto

[1] Vogliamo specificare che, considerata la natura della presente offerta, le nostre attivita' *non si spingeranno in nessun caso oltre questo punto (ESCALATING PRIVILEGES) a meno di una specifica autorizzazione in tal senso da parte del cliente.* In altre parole, si cerchera' di **dimostrare l'effettiva possibilità di assumere il controllo dei sistemi senza apportare alcuna modifica agli stessi.**

può diventare una "testa di ponte" per attaccare altre macchine. In tal caso saranno reperite informazioni riguardanti altri sistemi.

3.1.4.2. Covering traces and creating back doors

Prima di abbandonare il sistema "conquistato" vengono cancellati gli eventuali logs che hanno registrato la presenza clandestina ed eventualmente installati trojan o back-doors che consentano di rientrare facilmente sulla macchina in un secondo momento. Può essere utile anche installare tools nascosti quali sniffers o keyloggers al fine di catturare altre password del sistema locale o di altri sistemi ai quali utenti ignari si collegano dalla macchina controllata.

3.2. Assessment Applicativo

Questa analisi è costituita da una serie di tentativi d'attacco che coinvolgono solo i protocolli di comunicazione utilizzati dagli utenti finali per interagire con le applicazioni.
Nel caso specifico delle applicazioni web, tali attacchi sono basati su manipolazioni dei pacchetti HTTP che vengono scambiati fra i browser degli utenti ed il web server.
Esistono diverse categorie di attacchi verso applicazioni web, che possono portare alla compromissione di uno o più layer dell'intera infrastruttura applicativa: web server, application server, data tier.
Caratteristica comune a tutti gli attacchi applicativi è la completa trasparenza ad ogni sistema di difesa perimetrale (firewall, ids, ecc.): manipolazioni dei protocolli di layer 7 (applicativi) non possono essere rilevate da dispostivi che analizzano il traffico a layer 3 (network).
Il test sarà condotto in modalità anonima ed in "user-mode".
Ciò significa che, preventivamente, dovra' essere creato un account tramite le usuali procedure di attivazione al fine di permettere a Hacking Team di accedere come utente autorizzato. Non saranno accettati account di altro tipo (di test interno, amministrativi, etc.) poiché non fornirebbero la corretta valutazione circa il rischio che un utente registrato possa cercare di accedere in modo fraudolento ad informazioni per cui non è autorizzato.
L'attività comprende l'analisi dell'applicazione in termini architetturali, verranno analizzate le configurazioni delle macchine interessate, sia a livello di sistema operativo che applicativo.

In generale, le vulnerabilità di livello applicativo sono spesso legate ad errori contenuti nel codice delle applicazioni.

Esistono due classi di errori, che richiedono differenti strategie per essere identificati e rimossi: errori logico-architetturali ed errori di implementazione.

Errori logico-architetturali

Gli errori logico-architetturali consistono nel mancato utilizzo di meccanismi di sicurezza, oppure nell'utilizzo di meccanismi non adeguati a raggiungere lo scopo desiderato. Tali errori sono imputabili ad una non corretta definizione dei requisiti di sicurezza e/o ad una inadeguata progettazione dell'architettura.

Gli errori logico-architetturali più comuni sono i seguenti:

- gestione non corretta delle sessioni;
- uso di meccanismi di autenticazione deboli, che
    - permettono agli utenti di utilizzare password guessable;
    - rilasciano informazioni che permettono di restringere lo spazio di ricerca per attacchi di tipo brute force;
- trasmissione di informazioni sensibili su canali non cifrati;
- assunzioni errate in merito all'attendibilità e veridicità di input ricevuti dall'utente;
- assunzioni errate in merito alla funzionalità di sistemi e/o applicazioni client-side (ad esempio, browser web) che si trovano sotto il controllo dell'utente (o dell'attaccante!).

Errori implementativi

Questi errori si originano in fase di sviluppo, quando specifiche di alto livello, corrette dal punto di vista logico, vengono tradotte in codice che non gestisce correttamente tutti i casi i possibili; i malfunzionamenti che si verificano in casi particolari possono essere sfruttati per indurre nelle applicazioni comportamenti non previsti e/o non desiderati.

La grande maggioranza degli errori implementativi è dovuta ad una non corretta validazione dei parametri in ingresso, oppure alla gestione non corretta di alcuni input particolari, non previsti dal programmatore. La loro natura rende estremamente difficile prevederne l'impatto: in alcuni casi, questi errori possono avere conseguenze gravi sulla sicurezza di una applicazione, anche se gli elementi di codice interessati non sono direttamente legati a funzionalità critiche.

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.

Allo scopo di inquadrare il tema della sicurezza del livello applicativo, sia in termini di "opportunità" offerte all'intrusore, sia di minacce per le potenziali vittime di intrusioni, si dà una sintetica descrizione delle principali tecniche di attacco utilizzate nell'ambito delle applicazioni web.

I concetti e la terminologia introdotti saranno utilizzati nel presente documento per descrivere i risultati dell'assessment svolto.

3.2.1. Authentication brute-forcing

- Obiettivo: accesso aree riservate ad utenti in possesso di opportune credenziali.
- Attaccanti: chiunque non sia in possesso di credenziali valide ed abbia interesse ad accedere alle informazioni contenute nelle aree riservate, oppure chi, pur possedendo credenziali valide, intende accedere all'area riservata con l'identità di un altro utente.
- Descrizione: consiste nella sottomissione, spesso con l'ausilio di tool automatici, di un grande numero di credenziali (ad esempio coppie username,password), fino ad ottenere una risposta di autenticazione riuscita dal sistema. La generazione delle credenziali può prevedere l'uso di regole (ad esempio, generazione di tutte la password di sei caratteri costituite da soli caratteri alfanumerici) oppure di dizionari preesistenti.
- Condizioni necessarie per l'attacco: ogni sistema che dispone di un sistema di autenticazione è esposto a questo attacco.
- Probabilità di successo: dipende dalla dimensione dello spazio delle credenziali.
- Aspetti facilitanti: l'effort necessario per un attacco può essere sensibilmente ridotto da uno o più dei seguenti fattori:
  - password guessable: l'uso da parte degli utenti di password guessable aumenta la probabilità di successo degli attacchi basati su dizionario;
  - struttura delle credenziali: l'imposizione di una struttura semplice alle credenziali (ad esempio, password costituite da soli numeri o sole lettere, lunghezza non superiore a sei caratteri, ecc.) può diminuire sensibilmente la dimensione dello spazio di ricerca;
  - l'uso di messaggi di errore troppo informativi in caso di autenticazione fallita (ad esempio indicazioni che permettono di distinguere fra username errato e password errata) possono ridurre lo spazio di ricerca.

- Contromisure 2: gli attacchi di tipo brute force non possono essere prevenuti, ma esistono tecniche efficaci per ridurne drasticamente la probabilità di successo:
  - limitazione del numero massimo di tentativi di autenticazione falliti per ogni connessione;
  - adozione di controlli che vietano l'uso di password guessable o troppo semplici;
  - eliminazione dei messaggi di errori informativi.

3.2.2. Cross site scripting (XSS)

- 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. In alcuni casi particolari le stesse tecniche e le stesse vulnerabilità applicative possono essere sfruttate per provocare l'esecuzione di codice sul server (esempio: Server Side Include, ecc.). 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

[2] Le contromisure indicate in questo come in tutti gli altri casi sono da intendersi ovviamente come generiche. Caso per caso potranno essere o smentite, o confermate oppure rese più precise e puntuali.

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.

3.2.3. 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;
  - esecuzione di comandi di sistema tramite stored procedure.
- 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.

3.2.4. 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.

3.2.5. OS command injection

- Obiettivo: esecuzione di comandi di sistema sulle macchine su cui insiste l'applicazione.
- 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: questo attacco può essere effettuato quando la logica applicativa utilizza dati forniti in input dall'utente come parametri per l'esecuzione di comandi di sistema. Se la logica applicativa non esegue correttamente il parsing di tali dati, è possibile provocare l'esecuzione di comandi aggiuntivi e/o differenti da quelli previsti dagli sviluppatori.
- Condizioni necessarie: queste tecniche possono essere utilizzate in presenza di componenti dinamici che richiamano comandi di sistema senza effettuare un corretto parsing dei parametri di input.

- Probabilità di successo: dipendenti dall'accuratezza della logica di validazione dei parametri in input.
- Aspetti facilitanti: mancanza di funzionalità di filtraggio dell'output ritornato dopo l'esecuzione del comando di sistema.
- Contromisure: gli attacchi di questo tipo possono essere neutralizzati eliminando dai parametri in input token riservati e metacaratteri potenzialmente pericolosi che potrebbero generare ambiguità per l'interprete dei comandi.

3.2.6. 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.

3.2.7. 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.

3.2.8. Information leaking

- Obiettivo: ottenere informazioni sul sistema da attaccare.
- Attaccanti: chiunque possa navigare il sito.
- Descrizione: vengono esaminati i sorgenti HTML delle pagine web ritornate dall'applicazione, allo scopo di individuare informazioni sensibili, come
  - password cablate nel codice;
  - commenti erroneamente lasciati dagli sviluppatori;
  - informazioni su versioni del software utilizzato e configurazione.
- Probabilità di successo: dipendenti dal livello di security-awareness degli sviluppatori.
- Aspetti facilitanti: nessuno.
- Contromisure: eliminare dati sensibili dal codice HTML delle pagine web ritornate dall'applicazione.

3.2.9. Precisazioni

Nel caso in cui il test avvenga su ambienti in produzione occorre utilizzare, ove possibile, un account di test creato appositamente per la scansione.

- Per tale account devono valere le seguenti condizioni:
  - Accesso esclusivamente riservato a record di test nei database di back-end.
  - Ordini di acquisto o altre tipologie di transazioni dovrebbero essere ignorati.

  - Eventuali nuovi record creati da tale account devono essere successivamente cancellati.
  - Qualora le transazioni abbiano un qualche tipo di impatto (per esempio in caso di acquisto/vendita di azioni), il loro effetto dovrebbe riguardare esclusivamente dei record di test.
- Qualora l'applicazione preveda diversi livelli di privilegio, è consigliabile effettuare un'analisi con un numero di credenziali di test pari al numero dei profili esistenti e previsti.
- E' consigliabile preventivare e tenere in considerazione il tempo necessario allo sviluppo di script/procedure di clean-up per ripulire tutti i dati creati/modificati dall'utente di test.
- E' utile identificare e comunicare eventuali script o parametri che invalidino le sessioni al fine di evitare che durante le scansioni tali script o parametri vengano eseguiti dai tool di test automatizzati.
- Unitamente alla documentazione dell'analisi verrà rilasciato l'elenco delle URL sottoposte a scansione.

3.3. Criticità di un assessment

3.3.1. Denial of service

Il processo di eliminazione dei falsi positivi, dovuti ai *check* euristici effettuati dai sistemi di *vulnerability assessment* automatico, può richiedere il tentativo diretto di *exploiting* di un servizio, al fine di verificare l'effettiva presenza di una vulnerabilità, la possibilità di sfruttarla per prendere il controllo di un servizio, il suo impatto sulla sicurezza dei sistemi e dei dati in essi contenuti. Tuttavia, non tutte le classi di vulnerabilità richiedono l'utilizzo di *exploit* che possono compromettere la stabilità di un servizio o dell'intero sistema. Tipicamente, le vulnerabilità il cui tentativo di utilizzo può risultare in un D.o.S. sono quelle legate a problemi di *boundary check* e *integer overflow* (*stack/heap overflow*), *memory allocation*, *format string bug*, etc., il cui sfruttamento richiede la sovrascrittura di zone di memoria del processo contenenti strutture dati, indirizzi di ritorno, etc.

3.3.2. Perdita o inconsistenza di dati

Alcune classi di attacco applicativo (es: *SQL Injection*, *Cross Site Scripting*, etc.) prevedono l'accesso non convenzionale o la manipolazione di dati persistenti, tipicamente immagazzinati in un database relazionale. In alcuni casi, l'eliminazione dei falsi positivi (vedi punto precedente), o addirittura la semplice rilevazione della vulnerabilità, richiede la modifica permanente dei dati persistenti. Ad esempio, per verificare la possibilità di cancellare una tabella da un database SQL, sfruttando dei permessi d'accesso poco restrittivi o una mancata validazione degli input, è richiesta l'effettiva

cancellazione della tabella stessa. In altri casi, come ad esempio nelle vulnerabilità di tipo *Database Cross Site Scripting*, è richiesto l'inserimento di particolari *entries* malformate all'interno dei database utilizzati da un applicazione web-based. Questo potrebbe portare ad inconsistenze nel caso tali dati venissero utilizzati da un sistema di reportistica o *data-mining.*

3.3.3. File e processi zombie

Durante una simulazione d'attacco completa, alcuni tipi di approccio richiedono l'upload sulla macchina target di particolari tools (*netcat, pwdump,* etc.) o l'esecuzione di particolari processi, per permettere all'attaccante "simulato" di ottenere un pieno accesso alla macchina, per effettuare la cattura di dati sensibili o eliminare i log in maniera automatizzata, per elevare i propri privilegi, etc. In casi molto particolari non è possibile eliminare i file creati sulla macchina o i processi lanciati, senza un intervento diretto sul sistema da parte di un operatore.

3.4. Metodologia in caso di vincoli o divieti

Qualora i rischi connessi a particolari fasi dell'*assessment*[3] (eliminazione falsi positivi, simulazione d'attacco, etc.) non siano accettabili per il Cliente, è possibile ottenere i medesimi risultati utilizzando i seguenti tipi di approccio, unicamente a prezzo di una maggiore richiesta in termini di tempo:

- **Utilizzo sistemi di test:** E' possibile eliminare il rischio di potenziali disservizi causati dalle fasi di analisi più invasive (ad esempio i *Denial of Service* dovuti a tentativi di *exploiting* falliti), effettuando tali fasi sui sistemi di test. Questo tipo di attività deve essere preceduta da una verifica accurata dell'allineamento fra gli ambienti di test e di produzione. Nel caso non sia presente un ambiente di test, è possibile applicare delle procedure per ottenere una replica esatta dell'ambiente di produzione senza comprometterne l'operatività.
- **Verifica manuale:** Nell'eliminazione dei falsi positivi, in alternativa al tentativo diretto di *exploiting*, è possibile utilizzare un approccio di verifica manuale delle singole vulnerabilità rilevate dai prodotti di *assessment* automatico. Tale tipo di approccio prevede la verifica di presenza, e l'eventuale applicazione, di tutte gli aggiornamenti, *patch*, *best practice*, che possano eliminare o mitigare il problema riscontrato. Questa procedura, che deve essere comunque seguita per tutte le vulnerabilità che risultano effettivamente utilizzabili a scopi maliziosi, in questo caso deve essere applicata a tutte le criticità rilevate dai software di *scanning*.

[3] Le fasi più rischiose ed invasive di un assessment sono svolte unicamente qualora il Cliente richieda una particolare accuratezza nei risultati e nella valutazione degli scenari d'attacco e degli impatti. I rischi connessi ad *vulnerability assessment* "generico" sono in genere talmente bassi da essere accettabili per qualsiasi sistema che non sia considerato particolarmente critico.

- **Attacco manuale:** Negli attacchi di tipo applicativo è possibile eliminare o minimizzare il rischio connesso alla manipolazione dei dati persistenti non utilizzando software di analisi automatica. Le parti dell'applicazione che accedono a dati critici possono essere verificate manualmente eliminando il rischio di perdite accidentali, e permettendo l'immediato ripristino dei dati per cui è richiesta una manipolazione.

3.5. Tools utilizzati

Si elencano di seguito i tools che potrebbero essere utilizzati durante le attività. Si fa presente che il ruolo fondamentale in un'attività di assessment di qualsiasi tipo è dato dall'esperienza e dalla conoscenza di chi lo porta a termine; non è lo strumento che si utilizza che fa la differenza. Tant'è vero che abitualmente i tools automatici ricoprono solo una piccola parte (discovery e scanning) che è minimale rispetto al totale delle attività da intraprendere in un assessment professionale e di qualità.

3.6. Tools per l'assessment applicativo

I principali tools in questo ambito sono gli scanner applicativi: si tratta di tools che eseguono in modo automatico la navigazione (crawling) delle pagine web dell'applicazione target, riducendo significativamente il tempo necessario a ricostruirne la struttura completa. Tali scanner eseguono inoltre una serie di test finalizzati ad evidenziare l'eventuale vulnerabilità dell'applicazione ad una serie di attacchi comuni.

In alternativa esistono anche

- Web proxy: sono tool di intercettazione del traffico fra browser e server applicativo, che permettono di analizzare e modificare header e body di ogni singola richiesta/risposta HTTP.
- Decompilatori ed analizzatori di codice: per un'analisi approfondita dei binari che compongono la parte client-side dell'applicazione, vengono utilizzati dei software in grado di risalire a porzioni del codice sorgente originale, ed altri strumenti in grado di rilevare tracce di programmazione insicura.

Alcuni tra i tools utilizzati solitamente in questo ambito sono i seguenti:

- Domino Scan II - software vulnerability assessment per Lotus Domino
- NgsSquirrel - software vulnerability assessment per database (Oracle, DB2, SQL Server)
- WebInspect - WEB application assessment tool
- AppScan - WEB application assessment tool
- Paros - WEB Proxy
- Nikto - WEB Server assessment tool
- OraScan - Oracle WEB Application auditing

4. DOCUMENTAZIONE UTENTE

Oltre a quanto specificatamente richiesto nel capitolo 1 (RICHIESTA DEL CLIENTE), al termine dell'attività sarà fornito un report che conterrà:

a. **Topologia rilevata**
b. **Dettagliata descrizione del metodo e degli strumenti**
c. **L'elenco dei sistemi/apparati acceduti in modo non autorizzato**
d. **Descrizione della catena di eventi che hanno portato all'accesso della rete/sistema/applicazione**
e. **Log degli eventi**
f. **Eventuali esempi delle informazioni ottenute**

Sarà inoltre allegata una descrizione dei possibili miglioramenti che potrebbero essere applicati alla rete, ai sistemi o ai servizi, unita all'elenco, supra-vendor, delle soluzioni tecnologiche e/o dei prodotti da adottare per incrementare il livello di security del sistema informativo.

5. DOCUMENTI NECESSARI

Per dare inizio alle attività sarà necessaria la sottoscrizione dei due allegati:

- Allegato A: Accordo Legale (Liberatoria)
- Allegato B: Accordo di Non Divulgazione

6. RESPONSABILITÀ

Sarà responsabilità di HT completare il presente progetto secondo quanto specificato nella definizione delle funzionalità iniziali, fornendo al Cliente la documentazione citata.
Sarà responsabilità del Cliente garantire l'accesso ai locali preposti, nonché la disponibilità di una persona durante le attività previste dal presente progetto.
La presenza di tale persona permetterà a HT di spiegare nel modo più rapido ed efficace le attività svolte, sia in termini di tecniche che di strumenti.

7. OFFERTA ECONOMICA

Servizi	Descrizione	Costo
Ethical Hacking	**P.T. Applicativo**	**6.000,00**
Ethical Hacking	**P.T. Interno**	**6.400,00**
	Totale	**12.400,00**

I costi indicati si intendono al netto delle imposte.

8. CONDIZIONI GENERALI DI OFFERTA

Modalità di pagamento e condizioni generali di fornitura

Validità offerta	30 gg
Fatturazione servizi	100% all'ordine.
Liquidazione fatture	30 gg D.F.
Trasporti/Trasferta	Ns.carico
Garanzia	A norma di legge

Tutti i prezzi esposti nella presente offerta sono da intendersi IVA esclusa.

HT S.r.l.

Alessandro Lomonaco
Key Account Manager