Penetration Test: Punti di forza e di debolezza ed una possibile soluzione

Author : Fabrizio Baiardi

Date : 20 giugno 2018

Questo contributo cerca di illustrare alcuni punti di debolezza della valutazione della robustezza di un sistema basato su penetration test. Il nostro focus è ovviamente su debolezze strutturali del metodo di valutazione che, proprio perché strutturali, sono indipendenti dalle competenze e conoscenze dei team coinvolti nel test. Il penetration test è uno dei metodi più popolari per la valutazione della robustezza di un sistema rispetto alle intrusioni. L'idea di base è molto semplice: prendiamo qualcuno che sia, o dichiari di essere, un esperto di intrusioni e paghiamo questo tester per attaccare il nostro sistema, o meglio per ripetere le attività di un attaccante contro il nostro sistema. Se l'intrusione ha successo, il tester ci indicherà i difetti che ha sfruttato e noi potremo rimediare. In altre parole, il penetration test non è altro che il primo passo di una soluzione "penetrate and patch", trova i difetti per eliminali. Chiaro, semplice efficace. Come tutte le soluzioni semplici e risolutive, il diavolo sta nei dettagli. Avevamo già sollevato qualche dubbio in passato sulla complessità di scegliere le contromisure, si veda ICT Security del settembre 2015, che ora generalizzeremo.

Il principale punto di forza di un penetration test è la sua generalità, infatti esso può essere realizzato su un qualunque sistema dopo il suo deployment. Un qualunque sistema che può essere attaccato può essere il target di un penetration test. E'ovvio che si ricorre ad un penetration test in tutti quei casi in cui si ritiene sia impossibile misurare in modo "*ripetibile*" e "*condiviso*" la cyber security di un sistema. Con *ripetibile* intendiamo che la valutazione possa essere ripetuta da persone diverse in modo da verificarne i risultati. La ripetibilità richiede che il processo sia adeguatamente formalizzato definendo delle regole che indichino come gestire le possibili ambiguità che si possono incontrare durante l'attività. Con valutazione *condivisa* intendiamo che le caratteristiche del sistema da valutare e la scala di valutazione vengono concordati anticipatamente tra chi valuta e chi viene valutato. Quando non si conoscono metodi con queste proprietà ci si rivolge a prassi come il penetration test che permettono di avere comunque risultati in qualche modo utilizzabili. Nel seguito ci concentreremo su un black box penetration test, uno in cui l'attaccante non ha a propria disposizione informazioni sul sistema da attaccare ma deve scoprire tali informazioni durante il test. Questo è a nostro giudizio il caso più significativo perche rispetto ad altri tipi di penetration test abbiamo la massima similitudine

con un attacco reale. Raramente infatti un attaccante ha a disposizione informazioni sul sistema come avviene in un white box penetration test.

Il primo problema da risolvere per definire un penetration test è quello di trovare un esperto di attacchi informatici. Non ci sono molte certificazioni disponibili e mai come in questo caso il curriculum è fondamentale. Sfortunatamente il primo requisito per eseguire un penetration test è un dettagliato NDA che vieta di parlare di quanto fatto e dei risultati ottenuti. Superato questo problema, ad esempio ingaggiando un team con un currriculum significativo, dovremo cercare non un generico esperto ma un esperto con conoscenze e competenze adeguate per il nostro sistema. E'ovvio che utilizzare un esperto di Linux per attaccare sistemi Windows non sia una buona idea. La ricerca del penetration tester deve quindi comunicare all'esterno un numero di informazioni non banali sul nostro sistema.

Altri problemi sono il social engineering ed il phishing. Sono leciti o non lo sono? Il penetration tester può telefonare fingendo di essere la segretaria dell'amministratore delegato e dicendo che l'amministratore ha perso la password? Può mandare un interessante link che invogli a cliccare oppure no?

Una volta scelto il team e definite le regole del gioco, occorre decidere anche gli obiettivi dell'attaccante ovvero quali componenti, fisici o virtuali, deve controllare o quali informazioni deve esfiltrare o manipolare. A questo punto il penetration test, cioè l'attacco, può iniziare. Durante il penetration test, il sistema continua il suo normale funzionamento e gli strumenti di monitoraggio e difesa operano come al solito per supportare la normale gestione del sistema. Infine, i difensori, i.e. i gestori e gli amministratori del sistema non sono informati del test in modo da riprodurre con la maggior accuratezza possibile lo scenario dalla prospettiva sia dell'attaccante che del difensore.

Per continuare la nostra discussione, assumiamo che il sistema da penetrare sia di una ragionevole complessità e che quindi l'attaccante debba necessariamente costruire ed eseguire una catena di attacchi. Per costruire tale catena ogni black box penetration test deve comprende due tipi di attività che, per semplicità, indichiamo rispettivamente come *collect* ed *exploit*. L'obiettivo delle attività di *collect* è quello di raccogliere le informazioni sul sistema da attaccare. Infatti, visto che il test deve ripetere le attività di un attaccante, il penetration tester dovrà anche investire del tempo per raccogliere informazioni sui nodi del sistema target, le applicazioni che eseguono e le connessioni logiche e fisiche. Queste informazioni sono fondamentali per scoprire le vulnerabilità del sistema e quindi gli attacchi che possono essere efficaci. I meccanismi di base delle attività di *collect* sono il network scanning per scoprire i nodi di un sistema ed il vulnerability scanning per scoprire le loro vulnerabilità.

Le attività di *exploit* sono quelle che eseguono gli attacchi permessi dalle vulnerabilità scoperte dalle attività di *collect*. Metaforicamente, possiamo vedere le attività di *collect* come quelle che permettono all'attaccante di ricostruire una mappa dei vari componenti del nostro sistema e delle loro relazioni. Le attività di *exploit* sono quelle che utilizzano le informazioni raccolte per attaccare alcuni componenti, controllarli ed eventualmente usarli in altre attività di *collect.* L'attaccante sta esplorando un ambiente sconosciuto in cui deve attaccare alcuni elementi per usarli e raccogliere ulteriori informazioni per raggiungere il suo obiettivo.

Le due attività si alternano durante un test perché, ad esempio, è possibile scoprire i collegamenti di un certo componente solo dopo averlo attaccato con successo. Ad esempio, un attaccante può scoprire il sottosistema protetto da un firewall solo dopo aver attaccato con successo il firewall stesso. Nel penetration test di un sistema non banale, la percentuale di tempo necessaria per raccogliere informazioni non è assolutamente trascurabile e contribuisce significativamente al tempo necessario all'attaccante per raggiungere gli obiettivi concordati.

Ogni penetration tester deve, prima o poi nel suo attacco, risolvere quello che è stato chiamato il dilemma *collect or exploit*. Questo dilemma nasce quando l'attaccante può eseguire alcuni attacchi ma nessuno di loro permette di raggiungere l'obiettivo perché la mappa costruita fino a questo punto non permette di capire come raggiungere l'obiettivo. A questo punto l'attaccante può o eseguire uno degli attacchi possibili, sperando che i diritti acquisiti permettano di estendere la mappa e capire come raggiungere l'obiettivo oppure può raccogliere altre informazioni grazie alle quali estendere la mappa. La scelta ha ripercussioni non banali perché l'esecuzione di un attacco aumenta la probabilità che i difensori scoprano l'attaccante poichè la raccolta di informazioni richiede un tempo non banale e può generare un elevato carico di comunicazioni che, nuovamente, può permettere la scoperta dell'attacco. A conferma della complessità del dilemma possiamo ricordare come esista una letteratura con una grande quantità di proposte, teoriche e pratiche, per risolvere il dilemma, nessuna delle quali è però risolutiva.

Risolto l'amletico dilemma *collect or exploit*, l'attaccante sceglie un attacco in base ad un proprio insieme di preferenze e priorità e lo esegue. Solo se tale esecuzione ha successo, la costruzione della catena di attacco può proseguire. Altrimenti l'attaccante deve decidere come gestire il fallimento dell'attacco. Le soluzioni possibili sono la scelta di un altro attacco o la ripetizione dell'attacco fallito. Se non esiste un altro attacco, il penetration test può raccogliere altre informazioni, ad esempio mediante ulteriori vulnerability scanning per scoprire nuove vulnerabilità e quindi nuovi attacchi oppure dichiarare il fallimento del test.

La nostra breve descrizione evidenzia come ogni attaccante abbia una propria strategia che possiamo riassumere come la soluzione di tre dilemmi del penetration test:

1. Collect or exploit
2. Scelta di uno dei possibili attacchi
3. Gestione del fallimento di un attacco

Ovviamente, le singole soluzioni utilizzate e la loro fusione in una strategia complessiva hanno effetti significativi sul risultato del test ed anche sul tempo necessario all'attaccante per raggiungere gli obiettivi prefissati. Questo punto è di particolare rilevanza se il test fallisce. Infatti, a questo punto nasce un ulteriore e più importante dilemma ovvero **"un attaccante con una diversa strategia avrebbe avuto successo?".** Rispondere a questo ulteriore dilemma è l'unico modo di capire se il nostro sistema è davvero robusto oppure se non lo è ed ha resistito solo perché la strategia dell'attaccante è sbagliata. Sappiamo bene che ancora peggiore della mancanza di robustezza e sicurezza è la falsa sensazione di sicurezza e robustezza, quindi capire cosa sarebbe successo con un'altra strategia è davvero fondamentale per valutare la robustezza.

Raccogliere dati per risolvere il dilemma sulla strategia dell'attaccante *è però possibile solo se il penetration tester ripete il test con una strategia diversa*. Ma questo non è possibile perché per ripeter il test innanzitutto il tester dovrebbe dimenticarsi tutto quanto ha già scoperto sul nostro sistema come pure ripetere la telefonata in cui impersona la segretaria dell'amministratore delegato. Quindi possiamo rispondere alla domanda solo utilizzando un altro penetration tester che utilizzi una strategia diversa. Una prima deduzione è quindi che un penetration test possa essere ripetuto solo utilizzando non solo strategie diverse ma anche "tester" diversi. Anche in questo caso, un ulteriore fallimento resta comunque di difficile spiegazione perché non sapremo mai se attribuirlo a proprietà strutturali del sistema o ad eventuali carenze anche nella strategia del nuovo attaccante. In altri termini, possiamo cosi generalizzare il dilemma **"dopo quanti fallimenti di "penetration tester" diversi potremo affermare che il sistema è robusto?"**. Sottolineiamo che se, in passato, la probabilità di fallimento di un penetration test era molto bassa, il focus sugli aspetti di cybersecurity, i vincoli legislativi e gli investimenti previsti renderanno il fallimento del test un evento sempre più probabile. Quindi, in futuro, si incontreranno sempre più fallimenti e diventerà sempre più importante interpretarli e spiegarli. In termini statistici, la soluzione del dilemma sui fallimenti è un problema di confidenza che può essere cosi formulato "quale è il livello di confidenza sulla robustezza del sistema dopo k fallimenti di un penetration test". La statistica ci dice anche che l'unico modo di rispondere alla domanda sul livello di confidenza e poter interpretare i fallimenti del test è quello di ripetere più volte il penetration test, garantendo l'indipendenza di ogni esecuzione, cioè ristabilendo ogni volta le condizioni iniziali. In questo modo possiamo attribuire il fallimento a debolezze del sistema o dell'attacco. Una ripetizione indipendente può essere ottenuta solo automatizzando il test ovvero implementandolo mediante un programma. Ciò richiede una formalizzazione delle varie attività ancora più rigorosa ed estesa di quella precedente, in modo che essa possa essere affidata ad un programma. Il ripristino delle condizioni iniziali è in questo caso semplice, immediato e facilmente verificabile. Le strategie da applicare per risolvere i vari dilemmi possono essere modificate in fase di configurazione del programma. Se si dispone di questo strumento di attacco, che tipicamente utilizzerà meccanismi e politiche tipiche dell'intelligenza artificiale, è possibile ripetere più volte un test. Le ripetizioni utilizzano lo strumento nella stessa configurazione per valutare come eventi probabilistici influenzano il risultato del test. Ad esempio, ripetendo più volte il test possiamo analizzare la correlazione tra la probabilità di successo di uno specifico attacco ed il risultato finale del test. Invece, per scoprire come una specifica strategia influenzi i risultati del test, è necessario ripetere il test dopo aver modificato la configurazione dello strumento, e quindi la strategia dell'attaccante.

Una conferma indiretta sulla prospettiva della completa automatizzazione del penetration ci viene dai significativi investimenti che il DOD USA sta facendo sulla automatizzazione del processo che va dalla scoperta di una vulnerabilità all'attacco di un sistema. E' chiaro che se il sistema è complesso, l'attacco è in realtà una catena d'attacco la cui costruzione richiede l'adozione di una strategia che risolva i tre dilemmi del penetration tester prima delineati. Sempre più nel futuro il penetration sarà automatizzato e se, da un lato, questo permetterà di ripetere più volte il test, dall'altro pone il problema delle competenze necessarie non per attaccare un sistema ma per costruire, configurare e programmare un sistema che lo faccia in modo automatico. Ovviamente, i due insiemi di competenze hanno delle intersezioni ma sono fortemente diversi.

Tutto quanto detto fino ad ora, non risolve due ulteriori debolezze del penetration test, entrambe generate dal fatto che il test opera su un sistema esistente. Il primo problema di operare su un sistema esistente è il disturbo che il test genera. Per quanto il tester sia accorto e prudente, attività anche con un impatto relativamente basso come uno scanning possono spesso avere effetti non prevedibili su un sistema. Non è raro ad esempio che uno scanning di un sistema di controllo industriale possa rallentare in modo significativo il funzionamento del sistema con effetti a catena sul processo industriale che viene controllato. L'impossibilità di garantire l'assenza di tali effetti dovrebbe comunque sconsigliare l'adozione di questa soluzione in sistemi critici per la sicurezza dell'ambiente e delle persone come, ad esempio, sistemi di controllo del traffico o di impianti chimici potenzialmente inquinanti.

La seconda debolezza, a nostro giudizio la più significativa, è la possibilità di condurre il test solo dopo il deployment del sistema ovvero quando il sistema è stato progettato, installato e configurato. Ovviamente, ciò implica l'impossibilità di ottenere una "cyber security by design" e cioè di avere una qualche certezza sul sistema prima del suo utilizzo e, vero problema critico, riduce fortemente il numero di contromisure con un costo accettabile e proporzionato a disposizione dei difensori. Infatti, il costo di modificare strutturalmente il sistema in fase di progetto è di alcuni ordini di grandezza inferiore a quello di una modifica su un sistema già operativo. Questo problema non può essere risolto nemmeno automatizzando il penetration test ed affidandolo ad uno strumento. L'unica soluzione a questo problema è l'adozione di una tecnica di model based simulation in cui l'attacco viene condotto operando non sul sistema reale ma su un modello del sistema reale. Tale modello deve poter essere costruito sia quando il sistema è già esistente, analizzando i componenti utlizzati sia quando il sistema è ancora in fase di progetto, utilizzando le specifiche dei componenti che si prevede di utilizzare. Un ulteriore vantaggio di una soluzione model based è la possibilità di condurre valutazioni con gradi di accuratezza e costi diversi. Ciò è possibile modificando il livello di astrazione del modello utilizzato per il test. Una valutazione *quick and dirty* può utilizzare un modello molto semplice, una più accurata utilizzerà invece modello molto più fedele e dettagliato del sistema target.

Riassumendo, solo automatizzando il penetration test ed eseguendolo su un modello del sistema si possono superare i punti di debolezza prima evidenziati. Questa soluzione è in grado di offrire valutazioni accurate della robustezza di un sistema in tutta la sua vita, dalla progettazione al suo utilizzo in campo, fornendo inoltre informazioni statisticamente attendibili sulla resilienza del sistema, ad esempio dopo quanto tempo, quanti attacchi e con quale probabilità/livello di confidenza il sistema cominci a collassare. Se condotta in modo formale e ripetibile, questa valutazione può fornire inoltre informazioni fondamentali per la gestione dinamica del rischio ICT, la risposta ad attacchi e la gestione degli incidenti. Azzardando una previsione sul futuro, quella più difficile, possiamo dire che il futuro appartiene a chi sa automatizzare i penetration test.

A cura di: **F.Baiardi**, **J.Lipilini**, **M.Montecucco**, **F.Tonelli**
Università di Pisa e Haruspex srl