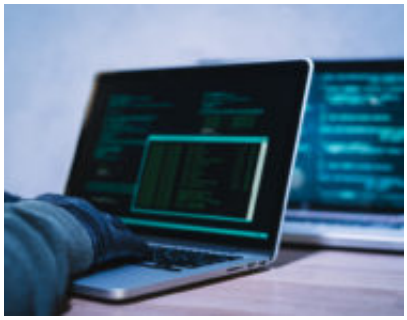
A case history: let’s rock!

Author : Vincenzo Digilio

Date : 24 Aprile 2019



Molti credono che la sicurezza informatica sia un ambito relegato alle grandi società, alle banche oppure a sempliciotti che pagano un riscatto in bitcoin per non vedersi pubblicati sul web in atti osceni[1]. La verità è che l’importanza della Cyber sicurezza è direttamente proporzionale alla nostra **dipendenza dalla tecnologia**, in parte “fisiologica” e dovuta all’evoluzione; essa però risulta anche inversamente proporzionale al grado di cultura di una società. La tecnologia si intreccia ormai con la nostra vita quotidiana in una sorta di simbiosi: informazioni, incontri, orari, stati d’animo, itinerari, domotica, automotive, sistemi elettorali, semafori, pacemaker, sistemi missilistici, ecc. Non vi stupirà quindi desumere le **implicazioni** che un virus informatico potrebbe potenzialmente avere, a partire dalla facoltà di danneggiare un organismo, compromettendo ad esempio le radiografie di un ospedale, sino al suo utilizzo per scopi politici[2].
Vi è mai capitato di ricevere pubblicità sul cellulare che vanno oltre la profilazione[3] derivata da ciò che digitate sui vostri dispositivi, ma anche dal vostro umore e/o magari da quello che avete detto?

Commissionato da terzi e in accordo con uno dei responsabili del Management, recentemente ci è stato richiesto un PenTest[4] per una società, ignara delle molteplici forme che un **attacco informatico** può assumere. Il responsabile sapeva il fatto suo ed era più che certo che la loro difesa fosse pressoché inespugnabile. Dietro firewall FortiGate, router Cisco e un Audit di Sicurezza sul sistema Active Directory Microsoft condotto da esperti, si sentiva all’apogeo della sicurezza informatica, un po’ come Priamo dietro le mura di Troia. Quindi, alla fine di un colloquio di circa quaranta minuti, si accomiatò con un commento del tipo «Sarà una “scansione di rete aggiuntiva…”». La frase fu proferita al limite tra la provocazione e il voler riportare un dato oggettivo, secondo il suo parere. È comune, infatti, pensare che un PenTest sia l’equivalente di Vulnerability Assessment con l’aggiunta di una successiva fase di Explotation[5]. In realtà, **un PenTest è molto di più**: esso è anche una ricerca di vulnerabilità *ex novo*, molte volte fatta dopo un’attenta analisi della rete, una selezione del target e ripetute azioni di testing su una determinata applicazione.

Il perimetro esterno si rivelò davvero ben protetto, era stato fatto un ottimo lavoro. Quindi avremmo dovuto trovare un'altra via d’ingresso. Decidemmo di procedere prima di tutto

con **un’analisi di Intelligence sul personale** della società. In particolare, notammo la propensione “social” di uno degli amministratori di rete, scoprimmo il suo debole per la musica e che faceva parte di una band rock. Utilizzava la pagina Facebook come propaganda per pubblicizzare le serate durante cui suonava. Fu semplice, a questo punto, ottenere l’amicizia dopo aver creato un profilo *ad hoc* per il nostro soggetto. Ovviamente la creazione del profilo fu stilata in modo da essere psicologicamente prossima al nostro Amministratore: ci fingemmo un fan e reperimmo ogni tipo d’informazione che aveva reso pubblica. Procedemmo scaricando una trial di trenta giorni del programma di cui si serviva l’azienda per la gestione e l’invio degli ordini, disponibile online dopo la compilazione di un comune form.

Il **Piano** che elaborammo fu, quindi, il seguente:

1. monitorare il profilo dell’amministratore di rete per stenderne una profilazione psicologica;
2. utilizzare la trial del programma appena scaricato come cavallo di Troia per entrare in possesso della rete.

Dovevamo quindi concentrarci sulla demo del programma scaricato, così lavorai per diversi giorni all’exploit dell’applicazione. Ciò che emerse fu la possibilità di un **buffer overflow**, vale a dire uno straripamento dei dati in ingresso in parti di memoria circostanti. Usando un gergo più tecnico, la stringa in input risulta più grande del *buffer* dove dovrà essere immagazzinata l’informazione. Questo porta a una sovrascrittura delle zone di memoria adiacenti al buffer, corrompendo e sovrascrivendo i dati in quel determinato settore. Spesso l’overflow produce un crash dell’applicazione, ma crea l’opportunità per l’attaccante di eseguire il codice arbitrario. Per essere più specifici, l’*overflow* si verificava al momento del login sull’app. Così, al fine di manipolare le zone di memoria, utilizzai il famoso Immunity debugger[6]. Poi agganciai la trial del programma al debugger, eseguendolo.

Spiegherò ciò che feci attraverso l’**esempio** che segue. Utilizzando il seguente simil-script in python:

```
#!/usr/bin/python buffer=["A"], counter=valore omesso,
 while,len(buffer),
```