Analisi di un firmware

Author : Gianluigi Spagnuolo

Date : 29 Aprile 2019

I firmware vengono utilizzati in contesti molto diversi tra loro: si va dai costituenti di un'infrastruttura industriale ai dispositivi del mercato *consumer*. In questo articolo vedremo **come analizzare un firmware** alla ricerca di possibili **vulnerabilità**.

Bisogna tener conto che in questi casi:

- la sicurezza non è un requisito, si preferiscono le prestazioni, un costo basso e delle dimensioni ridotte;
- una volta fatto il deploy, tipicamente, questi dispositivi rimangono in funzione (spesso senza aggiornamenti e manutenzione) per oltre 10 anni (la legge di Moore non ha valore).

Per la loro natura e per le loro capacità sempre maggiori, l'analisi di questi dispositivi è **sempre più complessa** e strutturata.

In genere, l'analisi non avviene sul codice sorgente ma sui file binari; sono diverse le motivazioni di questa scelta, ad esempio:

- sono coinvolte terze parti (consulenti, ricercatori) che non dispongono dei sorgenti;
- spesso viene commissionato un test *black box* (in pratica un test fatto con le stesse conoscenze reperibili da un attaccante).

Ci sono essenzialmente due tipi di firmware:

- **user-space firmware**: si ha quando è presente un sistema operativo *general-purpose* (come ad esempio Linux, FreeRTOS, VxWorks) e, quindi, molte funzionalità del firmware sono implementate in user-space. In questi casi, in fase di analisi, si può fare affidamento sulle primitive del sistema operativo e sulle funzionalità messe a disposizione dalle varie librerie.
- **Binary-blob firmware**: si ha quando il firmware viene eseguito direttamente sull'hardware del dispositivo, senza l'ausilio di un sistema operativo. È un caso

particolarmente complesso ma molto frequente. In questi casi, tipicamente, non si può fare affidamento sulla conoscenza del sistema operativo per un'analisi più agevole. Nella stessa situazione ci troviamo quando esiste un layer di astrazione dovuto ad alcune feature di un sistema operativo, ma sono soluzioni non-standard o non documentate; quindi, anche in questi casi la conoscenza di un OS e delle librerie non è d'aiuto.

Il risultato di queste problematiche è che l'individuazione delle vulnerabilità viene fatta manualmente. Analisi manuale che, oltre ad essere spesso anti-economica, può portare a diversi errori o approssimazioni dovuti alla complessità del codice in esame.

Schematicamente, i **passi** per un'analisi del firmware sono:

- Ottenere il firmware da analizzare;
- Fare un'analisi esplorativa di quanto ottenuto;
- Individuare i punti di interesse;
- Analisi: reverse engineering, penetration test, etc.;
- Emulazione: per confermare quanto scoperto in precedenza e per meglio comprendere il funzionamento del dispositivo.

Ottenere il firmware

Il firmware da analizzare si può ottenere in diversi modi, ad esempio:

- si può trovare nel materiale allegato al dispositivo da analizzare;
- se ne può fare il download dal sito dei produttori;
- si può trovare attraverso un motore di ricerca o nei siti aggregatori;
- si può ottenere dal traffico di rete estraendolo dai pacchetti intercettati durante un aggiornamento;
- si può ottenere facendo un *memory dump* del dispositivo, oppure, come ultima possibilità, facendo un *decap* dell'IC.

Analisi esplorativa

L'analisi esplorativa usa tecniche e strategie mutuate dal mondo dei dati, che risultano utili soprattutto quando abbiamo a che fare con un *blob* di dati e codice sconosciuto.

Se possibile, si possono ricavare informazioni utili semplicemente leggendo le specifiche e i *datasheet* distribuiti insieme al dispositivo. Capita spesso, però, che non siano disponibili né il *datasheet* né la documentazione; bisogna quindi estrapolare quante più informazioni possibile dal file binario, affidandosi a un'analisi esplorativa. Per prima cosa, bisogna trovare *indizi* per determinare il tipo di architettura; parliamo di *indizi* perché, almeno in questa fase, non si può avere ancora nessuna certezza riguardo al tipo di firmware in esame.

Questo tipo di analisi preliminare può essere fatta utilizzando diversi **strumenti**:

- l’utility *strings* o tool simili, come ad esempio *FLOSS* (https://github.com/fireeye/flare-floss);
- *Binwalk*: lo strumento che permette di analizzare un’immagine firmware e che si occupa di vari aspetti che vanno dalla scansione all’analisi dell’entropia;
- *hexdump* può essere usato sia per ispezionare l’header sia per individuare strutture/istruzioni di interesse;
- strumenti di machine learning: dalla classificazione per il riconoscimento dell’architettura costruita su feature basate su *opcode* (frequenza) fino all’individuazione di possibili vulnerabilità.

Questa prima analisi ci permette di fare delle assunzioni da verificare nelle analisi seguenti.

Analisi statica

Dopo aver individuato le aree di interesse, anche in funzione degli scopi dell’intera operazione, possiamo procedere con l’analisi vera e propria.

Dall’analisi esplorativa dovremmo avere, nel caso peggiore, almeno un’ipotesi su che tipo di firmware abbiamo di fronte. Prima di iniziare l’analisi vera e propria, occorre confermare queste ipotesi, in modo da identificare il tipo di firmware e prepararlo per le successive analisi.

Dalla definizione prima data, è subito chiaro che analizzare un firmware *user-space* è come analizzare un normale programma user-space e, quindi, non presenta particolari difficoltà. L’analisi di un *blob* binario invece è sicuramente più interessante e allo stesso tempo più impegnativa: sono ignoti e da scoprire l’indirizzo base e l’*entry point* del firmware.

Il **prodotto finale** di questa fase è un’immagine del firmware pronta per essere eseguita e analizzata.

In definitiva, l’analisi statica garantisce un’alta copertura e, allo stesso tempo, favorisce la precisione a discapito della scalabilità e della velocità di esecuzione.

In questo frangente, alcune ipotesi fatte rendono necessarie ulteriori verifiche. Inoltre, alcuni tool lavorano su rappresentazioni intermedie, aggiungendo un ulteriore livello di astrazione che, se da una parte è fondamentale per certe analisi, dall’altra non facilita la comprensione di un firmware.

Per sua natura, un’analisi statica presenta diverse approssimazioni: ad esempio, soffre della mancanza di informazioni derivate dall’esecuzione in ambiente reale, evidenti soprattutto quando abbiamo a che fare con un firmware.

Emulazione

L’analisi statica, come detto, presenta numerosi problemi, a volte capita che per venire a capo di alcuni elementi si prendano decisioni *audaci*, che possono portare a falsi positivi. Per ovviare

a ciò viene eseguita l’**analisi dinamica**, che analizza il comportamento del firmware mentre è in esecuzione. Un’analisi dinamica può essere vista come un modo per confermare quanto scoperto nell’analisi statica e definire nuove potenziali vulnerabilità.

Se da un lato è molto precisa e attendibile nei risultati, dall’altro la sua copertura può essere molto limitata, dipendendo dai path raggiungibili dagli input testati.

Nel caso di un firmware, il vero problema però è rappresentato dalla correttezza dell’emulazione. Può risultare complicato eseguire propriamente il firmware, con tutte le chiamate alle periferiche del dispositivo che si vuole emulare. Per questi dispositivi non è sempre semplice o possibile emulare completamente la realtà, tool come **Firmadyne** (https://github.com/firmadyne/firmadyne/), QEMU o anche emulatori commerciali hanno difficoltà ad emulare alcuni tipi di architetture.

Conclusioni

Abbiamo visto quali siano i passi per un’analisi completa di un firmware. L’ideale sarebbe automatizzare buona parte dell’intero procedimento - soprattutto le azioni ripetitive - in modo da rendere le analisi più efficienti in termini di tempo e copertura e di ridurre quanto più possibile il numero di errori.

Articolo a cura di **Gianluigi Spagnuolo**