Alma Mater Studiorum · Università di Bologna

CAMPUS DI CESENA
SCUOLA DI SCIENZE
Corso di Laurea in Scienze e Tecnologie Informatiche

FINESTRE DI VULNERABILITÀ E PACCHETTI SOFTWARE

Relatore:
Gabriele D'Angelo

Presentata da:
Giacomo Mantani

Correlatore:
Stefano Zacchiroli

Sessione II
Anno Accademico 2013–2014

PAROLE CHIAVE

Sicurezza Informatica
Vulnerabilità
Computer Forensics
Pacchetti Software

"We are never bored,
our childlike fascination for tecnology
keeps up busy" – (Anonimo)

Indice

Introduzione

La lingua italiana, rinomata in ambito letterario per scrittori e poeti, non si presta per l'esposizione delle tematiche in campo informatico. Questo testo è infatti ricco di inglesismi oramai diventati uno standard *de facto*. A questo proposito ho scelto di riportare tali parole direttamente in inglese, onde evitare traduzioni forzate che possano provocare confusione nel lettore. Termini stranieri e citazioni sono riportati in stile *corsivo*.

Nel corso di questa tesi si andranno a trattare diverse tematiche collegate alla sicurezza informatica ed in particolare alla misurazione, mediante l'utilizzo di standard internazionali, delle vulnerabilità che incorrono nella distribuzione Debian [1] GNU/Linux [2].
I primi due capitoli saranno discorsivi e permetteranno al lettore di comprendere il seguito del testo. Il primo capitolo tratterà della *computer forensics*, una disciplina che viene impiegata per analizzare le cause che possono aver prodotto un *computer security incident*. È un capitolo prettamente introduttivo, che non vuole certo essere esaustivo di questa tematica in realtà piuttosto vasta.
Il secondo capitolo tratterrà tutti gli standard che sono stati presi come riferimento al fine di sviluppare il software, che ha il compito di automatizzare una misurazione riguardante la sicurezza del sistema operativo all'interno di un intervallo di tempo limitato. Avere degli standard crea una conoscenza comune che permette di riuscire a tenere traccia dei rischi di sicurezza a li-

vello globale.

Dal capitolo tre si comincerà ad andare più sul tecnico, spiegando come Debian affronta i problemi di sicurezza, le misure che sono presenti e quelle che invece mancano e potrebbero essere sviluppate.

L'ultimo capitolo riprende parte del processo di sviluppo e di ingegneria del software, descrivendo una possibile soluzione nell'analizzare la storia dei pacchetti e delle vulnerabilità ad essi associate, ottenute tramite *database* liberamente accessibili in rete.

Capitolo 1

Computer Forensics

Forensics è il processo mediante il quale si utilizza la conoscenza scientifica per collezionare, analizzare e presentare fatti o dati alla Corte giudiziaria. Il termine inglese significa proprio *"portare alla Corte"*.
Nonostante sia una disciplina giovane, pur essendo largamente utilizzata da diversi anni in più ambiti, non vi sono metodologie standard tra Corti ed industrie nel mondo, dato dal fatto che la Legge è molto diversa tra gli Stati. Nel corso di questa tesi faremo sempre riferimento alla *Computer or Digital Forensics*, definita dal CERT [3] *"as the discipline that combines elements of law and computer science to collect and analyze data from computer systems, networks, wireless communications, and storage devices in a way that is admissible as evidence in a court of law."* [4].

Data la varietà di fatti e dati che potremmo avere, la *Digital Forensics* può essere utilizzata per diversi scopi, tra cui la ricostruzione di *computer security incidents*, che andremo proprio a trattare in questa tesi ed il cui scopo principale è quello di analizzare il sistema alla ricerca delle possibili cause che lo hanno portato ad una compromissione.

1.1 Metodologie di utilizzo

Esistono alcune fasi basilari che accompagnano ogni processo di *Digital Forensics*, che sono:

collezione delle informazioni dalle possibili sorgenti;

esame ed estrazione delle informazioni di interesse, preservandone l'integrità. Considerato il volume dei dati, generalmente questa fase è automatizzata e serve per filtrare solamente ciò che si vuole poi analizzare;

analisi delle informazioni che hanno passato la fase di estrazione per formulare le conclusioni riguardanti la procedura di *Digital Forensics* (generalmente la terminologia corrispondente è "*apertura del caso di analisi*");

report conclusivo o resoconto in cui vengono descritti gli aspetti utili a determinare le azioni da intraprendere per la risoluzione del caso con raccomandazioni utili ad evitare problemi futuri.

Ogni organizzazione dovrebbe avere la capacità di effettuare analisi di questo tipo, senza le quali sarebbe difficile determinare cosa e come si sono verificati incidenti di sicurezza all'interno di un sistema informatico o della rete.

Ulteriori dettagli su come integrare tecniche forensi ad eventuali incidenti sono accessibili nel documento del NIST *"Guide to integrating forensic techniques into incident response"* [5].

Capitolo 2

Standards

Una comunicazione di qualsiasi genere, per non incorrere in ambiguità, necessita di standard utilizzati per interpretarla.
Avremo a che fare spesso con alcuni dei principali standard in ambito security. In particolare:

- Common Vulnerabilities and Exposures 2.1
- Common Platform Enumeration 2.2
- Common Vulnerability Reporting Framework 2.3

2.1 Common Vulnerabilities and Exposures

CVE è una lista di *vulnerabilities and exposures* con l'obbiettivo di associare un nome proprio a problemi di sicurezza pubblici. In questo modo si evita di fare confusione all'interno di *tool*, *repository* e servizi quando ci si riferisce a specifici problemi legati alla sicurezza.
Con il termine vulnerabilità si indica quindi un errore legato generalmente al software, che può essere utilizzato da un malintenzionato per ottenere accesso a sistemi o reti altrui.
Le *"esposizioni"* sono degli errori nel software che permettono l'accesso non autorizzato ad informazioni inerenti sistemi e reti, che possono portare alla

scoperta o all'utilizzo di una particolare vulnerabilità.
Questa lista è stata creata dal MITRE [6], in collaborazione con Governi, industrie, mondo accademico, istituti di ricerca ed esperti nel settore, con l'intento di migliorare la *"misurabilità"* della sicurezza. Si tratta di una lista pubblica accessibile a chiunque sia interessato a scaricarla, redistribuirla, farne riferimento o analizzarla, ma senza modificarne le informazioni in essa contenute. È possibile fare richiesta per aggiungere una nuova segnalazione, previa approvazione del MITRE.
Il processo per la creazione di un nuovo CVE comincia con la scoperta di una potenziale vulnerabilità o *exposure*, a cui viene assegnato un identificatore chiamato *CVE-ID, CVE Entry, CVE Name* o semplicemente *CVE* da un authority CVE Numbering Authority (CNA) una volta verificato.

2.1.1 Formato identificatore

Il formato dell'identificatore dei CVE sta cambiando la sua sintassi dato l'aumento di vulnerabilità segnalate ogni anno. Il vecchio formato aveva una sintassi del tipo CVE-YYYY-NNNN dove YYYY era l'anno di pubblicazione e NNNN un numero incrementale che quindi poteva coprire un massimo di 9999 vulnerabilità annue. La nuova sintassi CVE-YYYY-NNNN[N]{0,3}, con un numero arbitrario di N che varia da 4 a 7, permetterà di tracciare più di 10000 vulnerabilità annue e verrà cambiata ufficialmente il 1 Gennaio 2015. Tutti i precedenti CVE-IDs, prima dell'avvenuto passaggio di sintassi, rimarranno inalterati in quanto comunque compatibili con quella nuova.

2.1.2 Informazioni sul contenuto

Data la vastità di casi che possono suscitare l'interesse di creare una nuova segnalazione, un problema può essere suddiviso in più CVE-ID invece che in uno unico.
Esistono delle linee guida che facilitano la scelta di scissione o meno e che si possono trovare all'indirizzo web [7].

Maggiori informazioni riguardo la lista dei CVE sono accessibili online all'indirizzo web [8] e per una completa spiegazione delle terminologie vedere [9].

2.2 Common Platform Enumeration

Un Common Platform Enumeration (CPE) è uno standard per descrivere ed identificare classi di applicazioni, sistemi operativi e dispositivi hardware all'interno di un sistema informatico. L'identificativo è legato alla coppia nome-versione, dove il nome spesso rimane fisso durante lo sviluppo, mentre la versione cambia in funzione delle variazioni apportate.
Nel caso di un software abbiamo un CPE differente per ogni aggiornamento. Ricollegandoci ai Common Vulnerabilities and Exposures (CVE), spiegati nella sezione precedente, ogni *advisory* (avviso) sarà associato ad uno o più CPE. Per ulteriori informazioni visitare [10].

2.3 Common Vulnerability Reporting Framework

Common Vulnerability Reporting Framework (CVRF) è un documento XML che permette la condivisione di informazioni generiche riguardanti la sicurezza, quindi non solamente vulnerabilità, ma ogni documentazione collegata a questa tematica. È stato creato come un vero e proprio *framework* per la stesura di *security-reports*, con l' obbiettivo di definire un modello di facile lettura, condivisione e diffusione.
La lista dei CVE può essere scaricata in questo formato [11] ed è possibile trovare ulteriori informazioni sullo standard CVRF all'interno del sito web dell'ente incaricato alla manutenzione [12].

Capitolo 3

Sicurezza in Debian

Debian affronta i problemi di sicurezza con molta attenzione utilizzando una "politica aperta", con la quale si intende l'esposizione al pubblico di tutti i problemi presenti e noti all'interno della distribuzione (fonte principale [13]). Citando il **Debian Social Contract** [14]:

"We will not hide problems.
We will keep our entire bug report database open for public view at all times.
Reports that people file online will promptly become visible to others."

Oltre ad essere visibili online, i problemi sono resi pubblici e discussi con la *community* mediante la lista di distribuzione *debian-security* [15], nella quale sono pubblicate tutte le Debian Security Advisory (DSA) (vedi sezione 3.1). Questa possibilità di avere tutto esibito permette di arrivare ad una soluzione dei problemi in tempi relativamente brevi.
Il rilascio degli aggiornamenti viene applicato a tutte le architetture supportate di ogni release: *stable*, *sid* (detta anche *unstable*) e *testing*. Ulteriori informazioni sulle tipologie di *release* è possibile trovarle nella documentazione online [16].
Tutte le informazioni riguardo la sicurezza sono centralizzate [17] ed esiste un manuale di riferimento [18].
È possibile contattare il team sicurezza, oltre che tramite la *mailing-list*,

tramite il canale Internet Relay Chat (IRC) #debian-security accessibile nel server irc.oftc.net.

3.1 Debian Security Advisory - DSA

Le Debian Security Advisories (DSA) sono create ogni volta che viene scoperta una vulnerabilità riguardante Debian e vengono verificate da un membro del team di sicurezza. Al loro interno troviamo informazioni come:

- descrizione del pacchetto;
- numero della versione aggiornata (in cui i problemi sono stati risolti);
- tipologia del problema;
- riferimento ai CVE associati;
- descrizione di come viene sfruttata la vulnerabilità (*exploit*);
- se è possibile sfruttarla localmente o da remoto;
- descrizione della soluzione trovata;
- eventuali note.

Il formato utilizzato è simile a quello visto per i CVE 2.1: DSA-NNNN dove NNNN rappresenta un numero incrementale.

Come sopra accennato, per ogni segnalazione vengono generalmente inclusi anche i CVE associati. Questo offre diversi vantaggi come la possibilità di ricondurre vulnerabilità generiche a specifici aggiornamenti o di utilizzare software Intrusion Detection System (IDS) o di *Vulnerability Assessment* sviluppati in contesti non necessariamente in relazione con Debian.

Esistono casi in cui un CVE può non essere presente nelle segnalazioni DSA, tra i quali abbiamo:

- nessun pacchetto Debian è affetto dalla vulnerabilità a cui fa riferimento il CVE;
- non è ancora stata pubblicata la segnalazione DSA. Il bug potrebbe essere noto ma non è ancora pronto un aggiornamento;
- la segnalazione è stata pubblicata prima che il CVE sia stato ufficialmente assegnato, per motivi di *disclosure* o semplicemente perché è stato trovato dal team di sicurezza Debian che nello stesso momento in cui ha creato la segnalazione DSA ha richiesto il rilascio di un nuovo CVE.

Dal 24 Febbraio 2004, la lista DSA viene dichiarata ufficialmente CVE compatibile [19].

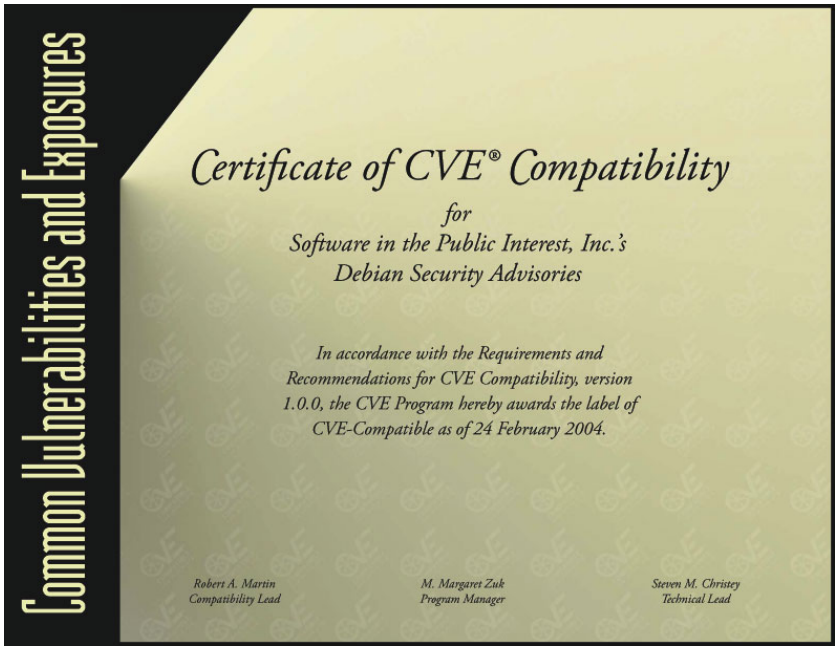

Figura 3.1: Certificato ufficiale compatibilità CVE

3.2 Secure Testing Repository

Il sistema di *tracking* [20] per i bug e le vulnerabilità è basato su un *repository* Apache Subversion (SVN) composta per la maggioranza da file di testo e *script* utili per generare un bollettino di sicurezza accessibile online.

Il modo migliore per rendersi conto del contenuto è scaricare direttamente il *repository*, seguendo due alternative:

- come utenti autenticati Alioth
  `svn co svn+ssh://<al.name>@svn.debian.org/svn/secure-testing`
- come utenti anonimi, senza permesso di scrittura
  `svn co svn://anonscm.debian.org/svn/secure-testing`

Eventualmente è possibile accedere anche ad una versione online, sempre solo in lettura, tramite il servizio viewvc [21] all'indirizzo https://anonscm.debian.org/viewvc/secure-testing/.

Il nome della cartella "secure-testing" è mantenuto per motivi storici e non vuole fare riferimento alla *release* "testing".

All'interno abbiamo diverse sotto cartelle di particolare interesse:

bin : contiene tutti gli script utilizzati all'interno del *repository* per la gestione del database delle vulnerabilità, la generazione di nuovi *advisory* e l'utilizzo del servizio di *tracking*;

lib : contiene i moduli python utilizzati all'interno dei vari progetti; i più importanti sono "debian_support.py" e "bugs.py", il primo implementa facilitazioni nel manipolare metadati Debian, come possono essere quelli associati ad un pacchetto, il secondo invece aiuta la gestione di file come "DSA/list" e "CVE/list";

data : contiene i dati veri e propri, tra cui tutte le liste di vulnerabilità come "CVE/list" e "DSA/list". I file "list", di cui parleremo più in dettaglio a seguire, vengono aggiornati due volte al giorno da un *cronjob*.

3.2.1 Livelli di sicurezza

I livelli di sicurezza sono usati soprattutto per indicare, data una sequenza di problemi, quali siano quelli di maggiore interesse da risolvere rispetto agli altri. Abbiamo:

unimportant Problemi che influenzano sorgenti, quindi non pacchetti *binari* (già compilati). Oltre ai file sorgenti fanno parte di questa categoria anche i file di esempio ("doc/foo/examples/*") e tutte quelle problematiche non realmente realizzabili nella pratica (*"non-issues in practice"*);

low Denial of Service attack (DoS) locali e "/tmp" file *races*.

medium Vulnerabilità che permettono esecuzione di codice tramite l' interazione con l'utente come attacchi di *privilege escalation* locali o remoti

(se limitate all'applicazione e quindi senza accesso shell) e Denial of Service attack (DoS) remoti;

high Tipico problema che **necessita di essere risolto** o raggirato. Generalmente fa riferimento a codice largamente diffuso (*attack vector* molto ampio), con un *exploit* associato disponibile. Si inserisce inoltre a questa categoria ogni vulnerabilità che permette ad un attaccante di eseguire codice arbitrario in un sistema remoto (con o senza privilegi di root) e tutti gli attacchi DoS con un forte impatto (come *IPv4 forwarding path vulnerability* che richiede pochi pacchetti per esser sfruttata). Possono inoltre venire aggiunte problematiche che producono difetti significativi a comportamenti software, come malfunzionamenti in algoritmi crittografici.

3.2.2 CVE/list

Questa lista, seppur popolata mediante le segnalazioni presenti all'interno dell'archivio ufficiale, (scaricabile in diversi formati [22]) è automaticamente modificata da alcuni script interni che aggiungono informazioni utili al team di sicurezza, il cui compito è quello di controllare tutte le segnalazioni contrassegnate dal nome "TODO" per stabilire se interessano Debian ed in particolare quale pacchetto e con quale livello di sicurezza. Per farlo è necessaria una ricerca approfondita che spesso richiede di leggere sorgenti e controllare se non sono già presenti aggiornamenti in fase di elaborazione. Oltre le segnalazioni da controllare abbiamo informazioni riguardanti punti descritti a seguire.

Segnalazioni non inerenti

Sono tutte quelle segnalazioni inerenti a moduli di terze parti che non influiscono e non sono collegati con Debian, contrassegnate come "NOT-FOR-US" (NFU).

Esempio:

```
CVE-2005-3018 (Apple Safari allows remote attackers to ...)
NOT-FOR-US:Safari
```

Se si vuole evidenziare ed ordinare tutte le segnalazioni NFU, è possibile utilizzare lo script all'interno del *repository* `"bin/check-new-issues"`.

Segnalazioni riservate

Diversi problemi di sicurezza ottengono un CVE prima della loro data di pubblicazione (*disclosure*). In questo caso, esiste un CVE-ID, ma non sono ancora pubblici i dettagli della vulnerabilità, per cui la segnalazione viene contrassegnata come "RESERVED", esempio:

```
CVE-2005-1432
RESERVED
```

Segnalazioni rifiutate

Tutte le segnalazioni che risultano essere duplicate o sbagliate sono contrassegnate come "REJECTED", esempio:

```
CVE-2005-4129
REJECTED
```

Segnalazioni senza un CVE-ID

Se si viene a scoprire una vulnerabilità, prima che sia reso pubblico il CVE, si può aggiungere una segnalazione.

Esempio:

```
CVE-2009-XXXX [optipng array overflow]
- optipng 0.6.2.1-1 (low)
NOTE: http://secunia.com/advisories/34035/
```

Da come si può notare, è preferibile aggiungere un riferimento al Bug Tracking System [23], in modo da aiutare il responsabile allo sviluppo del pacchetto affetto.

Informazioni addizionali

Spesso capita che il lavoro di ricerca da fare per una segnalazione richieda più tempo del dovuto e non si riesca a portare a termine. In questo caso si può aggiungere una riga "TODO" nelle note, seguita da una descrizione in modo tale che siano chiare per gli altri utenti le azioni rimaste in sospeso. Esempio:

```
CVE-2005-3990 (Directory traversal vulnerability in FastJar 0.93
allows remote ...)
TODO: check, whether fastjar from the gcc source packages is
affected
```

Nelle righe per le note, potrebbe essere utile anche aggiungere chiarimenti e/o riferimenti pur rimanendo il più sintetici possibile, un esempio:

```
CVE-2005-3258 (The rfc1738_do_escape function in ftp.c for Squid
2.5 STABLE11...)
- squid <not-affected> (bug #334882; medium)
NOTE: Bug was introduced in a patch to squid-2.5.STABLE10,
NOTE: this patch was never applied to the Debian package.
```

3.2.3 DSA/list

È una lista contenente tutti gli advisory trovati che riguardano le *release stable* e *oldstable*. Un esempio di segnalazione DSA è la seguente:

```
[21 Nov 2005] DSA-903-1 unzip - race condition
CVE-2005-2475
[woody] - unzip 5.50-1woody4
[sarge] - unzip 5.52-1sarge2
NOTE: fixed in testing at time of DSA
```

La prima riga tiene traccia della data di stesura, dell'identificatore, del pacchetto affetto e del tipo di vulnerabilità. La seconda contiene l'elenco degli id per i CVE associati, che vengono poi aggiornati automaticamente mediante l'uso di uno script. La riga successiva contiene gli aggiornamenti per la *release stable* ed eventualmente viene aggiunta un ulteriore riga per la *release oldstable*. Possono essere aggiunte, inoltre, delle note in chiusura, generalmente utilizzate per scopi statistici.

3.3 Debian Security Analyzer

Debian fornisce diversi software per monitorare e mantenere in sicurezza il sistema [24]. Risulta di particolare interesse, per quanto riguarda la seguente trattazione, il *tool* denominato "Debian Security Analyzer", abbreviato debsecan.
Il programma valuta lo stato di un *host* evidenziando se sono presenti vulnerabilità ed aggiornamenti tra i software installati. Le informazioni riguardanti lo stato corrente dei programmi vengono fornite dal file `/var/lib/dpkg/status` e vengono messe in relazione con l'elenco di vulnerabilità accessibili tramite il servizio *Debian Security Tracker* [20], parte del progetto gestito dal *Debian Security Team* team@security.debian.org.

3.3.1 Funzionalità

Le funzionalità si possono intuire facilmente tramite l'output che si ottiene eseguendo il comando `debsecan --help`:

```
Usage: debsecan OPTIONS...

Options:
  -h, --help          show this help message and exit
  --config=FILE       sets the name of the configuration file
  --suite=SUITE       set the Debian suite of this installation
  --source=URL        sets the URL for the vulnerability information
  --status=NAME       name of the dpkg status file
  --format=FORMAT     change output format
  --only-fixed        list only vulnerabilities for which a fix is
  available
  --no-obsolete       do not list obsolete packages (not recommend)
  --history=NAME      sets the file name of debsecan's internal
  status file
  --line-length=LINE_LENGTH
                      maximum line length in report mode
  --update-history    update the history file after reporting
  --mailto=MAILTO     send report to an email address
  --cron              debsecan is invoked from cron
  --whitelist=NAME    sets the name of the whitelist file
  --add-whitelist     add entries to the whitelist
  --remove-whitelist remove entries from the whitelist
  --show-whitelist    display entries on the whitelist
  --disable-https-check
                      disable certificate checks
  --update-config
```

Bisogna precisare che *suite* sta ad indicare la *release* e bisogna specificarla sempre quando si utilizza l'opzione `--only-fixed`. Per maggiori dettagli,

consultare il manuale `man debsecan`.

Le informazioni sulle vulnerabilità sono scaricate di default all'indirizzo https://security-tracker.debian.org/tracker/debsecan/release/1/GENERIC. `GENERIC` viene poi sostituito con il nome della *release* indicata tramite il comando `--suite`. Per avere un idea sul contenuto basterà scaricare l'archivio (in formato zlib), utilizzando ad esempio `GNU wget` [25], per poi utilizzare uno script Python simile al seguente per decomprimere l'archivio e redirigere l'output in *stdout*:

Vedi pagina seguente.

```
from zlib import decompress
from cStringIO import StringIO
import sys
import argparse
# Command line parser
def parse_cli():
    """Reads sys.argv and returns an options object."""
    parser = argparse.ArgumentParser(description='Decompress
 → zlib archive')
 → parser.add_argument("archive", help="Zlib compress
 → data file")
    return parser.parse_args()
def main(args):
    data = []
    try:
        f = open(args.archive, 'rb')
    except IOError:
        sys.exit(1)
    while 1:
        d = f.read(4096)
        if d:
            data.append(d)
        else:
            break
    print StringIO(decompress(''.join(data))).readlines()
if __name__ == '__main__':
    main(parse_cli())
```

L'output può essere di diversi formati, scelti mediante il comando `--format`. Vediamo un esempio del formato di default, *summary* :

```
['VERSION 1\n', 'CVE-1999-0559,,A system-critical Unix file or
```

directory has inappropriate ...\n', 'CVE-1999-0710,,The Squid package in Red H at Linux 5.2 and 6.0, and other ...\n', 'CVE-1999-0997,,wu-ftp with FTP conversion enabled allows an attacker to execute ...\n', 'CVE-1999-13 32,,gzexe in the gzip package on Red Hat Linux 5.0 and earlier allows ...\n', 'CVE-1999-1572,,cpio on FreeBSD 2.1.0, Debian GNU/Linux 3.0, an d possibly other ...\n', 'CVE-2000-0692,,ISS RealSecure 3.2.1 and 3.2.2 allows remote attackers to cause a ...\n', 'CVE-2000-0992,,Directory traversal vulnerability in scp in sshd 1.2.xx allows a ...\n', 'CVE-2000-1219,,The -ftrapv compiler option in gcc and g++ 3.3.3 and earlier d oes not ...\n', 'CVE-2000-1220,,The line printer daemon (lpd) in the lpr package in multiple Linux ...\n', 'CVE-2000-1221,,The line printer d aemon (lpd) in the lpr package in multiple Linux ...\n', 'CVE-2000-1226,,Snort 1.6, when running in straight ASCII packet logging mode or IDS ...\n', 'CVE-2000-1247,,The default configuration of the jserv-status handler in jserv.conf in ...\n', 'CVE-2001-0131,,htpasswd and htdigest in Apache 2.0a9, 1.3.14, and others allows local ...\n', 'CVE-2001-0735,,Buffer overflow in cfingerd 1.4.3 and earlier with the ...\n', 'CVE-2001-0775,,Buffer overflow in xloadimage 4.1 (aka xli 1.16 and 1.17) in Linux ...\n']

3.3.2 Implementazione

Debsecan è un software python composto da un solo file per scelte implementative di portabilità. Al suo interno utilizza la libreria `python-apt` [26] composta da due moduli, apt_pkg e apt_inst scritti sulla base delle corrispondenti librerie C++. Sempre per motivi di portabilità contiene al suo interno parte del sorgente "debian_support.py" tra cui le classi descritte nel diagramma che segue.

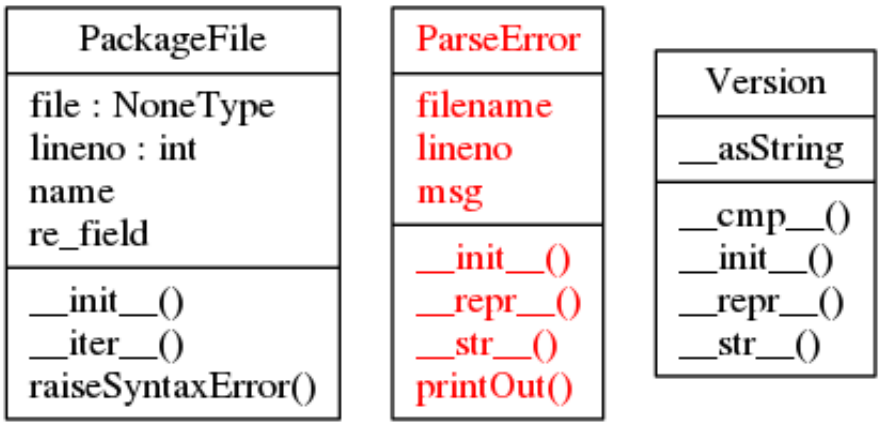

Figura 3.2: Classi importate da Debian Support

3.3.3 Limitazioni

Il Debian Security Analyzer è molto utile quando si vuole mantenere in sicurezza il sistema in operazioni di routine che precedono e cercano di evitare un potenziale attacco. Quando però la macchina è già stata compromessa, si è interessati maggiormente ad analizzare le potenziali cause che hanno reso l'attacco realizzabile. Lo stato corrente dei programmi passa in secondo piano rispetto alla loro storia, intesa come intervallo di tempo limitato durante il quale vi sono stati o meno aggiornamenti, installazioni o rimozioni di applicazioni. Riuscire ad analizzare la storia di un sistema è una funzionalità che non è ancora stata implementata in debsecan.

Capitolo 4

Lapsus

Lapsus è il nome del software sviluppato per cercare di portare l'analisi effettuata da debsecan ad un livello più elevato. L'obbiettivo è quello di valutare lo stato dei pacchetti installati, indicandone le finestre di vulnerabilità, ovvero gli intervalli di tempo a cui ognuno di essi è rimasto affetto.

4.1 Requisiti

Per la stesura dei requisiti si è preso come riferimento lo standard [27]. Si richiede la compilazione di un report che contenga una lista di intervalli di tempo (*time windows*) in cui si descriva, per ogni pacchetto software installato, le vulnerabilità pubblicamente note ad esso associate in quel momento.

Obbiettivo	Stimare, mediante l'analisi dei file di log del *package manager*, le vulnerabilità a cui il sistema era soggetto in un determinato arco di tempo.
Scope	GNU/Linux Debian, *package manager* dpkg.
Attori	Utente, advisory remote.

Precondizioni	Connessione ad internet.
	Presenza ed accesso al file di log, considerato integro e non compromesso.
	Non si possono fare previsioni euristiche, per mancanza di dati, quindi il periodo di tempo arriva ad analizzare al più l'odierno.
Successo esecuzione	Resoconto reale che evidenzi solo se son presenti vulnerabilità durante l'intervallo di tempo preso in analisi.
Insuccesso esecuzione	Arresto imprevisto del sistema.
	Repository online irraggiungibile.
	Log assenti, compromessi o inaccessibili.
Post-condizioni	I risultati sono da considerare solo un aiuto all'analisi. Non si hanno certezze sulle *reali* vulnerabilità che possono essere presenti ma non ancora di pubblico dominio.

4.2 Analisi dei Requisiti

Il tool sviluppato si basa sulla distribuzione Linux Debian ed in particolare sul software di *package management* dpkg ed è configurabile su più aspetti che saranno trattati nel dettaglio a seguire.

4.2.1 Scenario

Segue il diagramma di un caso d'uso tipico. Per lo use case template si è preso come riferimento lo standard di Alistair Cockburn [28].

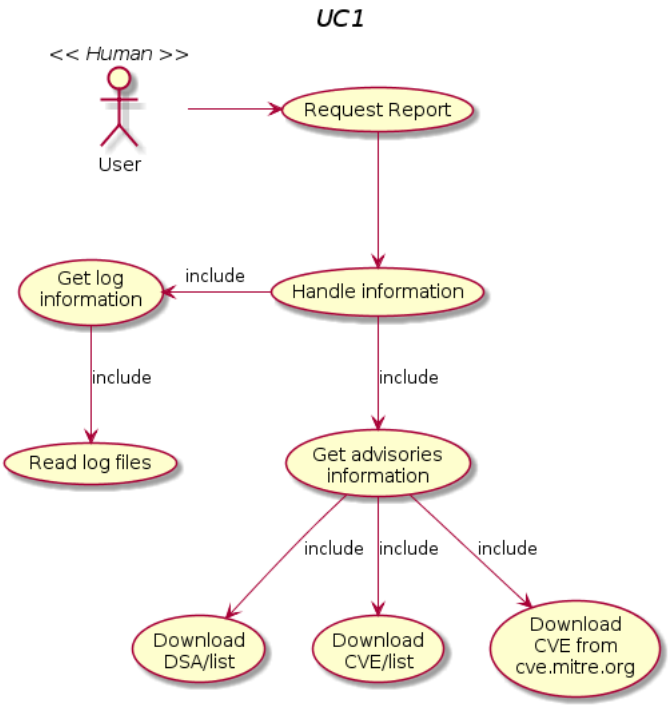

Figura 4.1: Diagramma dei casi d'uso

4.3 Analisi del Problema ed implementazione

Per la parte implementativa si è ereditata la struttura del software debsecan, rispettandone la licenza e cercando di introdurre, oltre alle funzionalità già esistenti, una parte che andasse a risolvere i requisiti richiesti elencati nelle sezioni precedenti 4.1 4.2.
L'obbiettivo è sempre stato quello di cercare di mantenere il più possibile le linee guida della filosofia di sviluppo Debian e le scelte implementative di *Florian Weimer*, fautore del tool debsecan.
L'implementazione è stata svolta in funzione di tre importanti fasi:

- costruzione della finestra temporale, ovvero storia di tutti i pacchetti che sono stati installati, rimossi e/o aggiornati;

- acquisizione di tutti i possibili aggiornamenti di sicurezza per relazionare il tempo di esposizione ad una vulnerabilità, nonostante fosse disponibile una sua *patch*;

- acquisizione di tutte le vulnerabilità di cui non si era a disposizione di un aggiornamento per evidenziare il tempo di esposizione (inevitabile) ad attacchi.

4.3.1 Costruzione finestra temporale

Per costruire una finestra temporale sulla storia di tutti i pacchetti software, bisogna ottenere informazioni riguardo lo stato corrente e le modifiche (installazioni, aggiornamenti, rimozioni) avvenute a partire dall'inizio del periodo che vuole essere preso in esame. Sia le informazioni riguardo lo stato corrente che le modifiche, sono presenti in file di testo gestiti automaticamente da dpkg. Il percorso di *default* è per lo stato corrente /var/lib/dpkg/status e per le modifiche /var/log/dpkg.log[.N]. I log possono essere suddivisi in più file nel caso in cui siano sottoposti a tool di *rotation* (come logrotate) in modo da ridurre l'occupazione su disco mantenendo versioni separate compresse in archivi dati.

Esempio di informazione contenuta nel file /var/lib/dpkg/status:

```
Package: vim
Status: install ok installed
Priority: optional
Section: editors
Installed-Size: 2063
Maintainer: Debian Vim Maintainers ...
Architecture: amd64
Version: 2:7.4.430-1
Provides: editor
Depends: vim-common (= 2:7.4.430-1), vim-runtime (= 2:7.4.430-1)...
```

```
Suggests: ctags, vim-doc, vim-scripts
Description: Vi IMproved - enhanced vi editor
 Vim is an almost compatible version of the UNIX editor Vi.
 .
 Many new features have been added: multi level undo, syntax
 highlighting, command line history, on-line help, filename
 completion, block operations, folding, Unicode support, etc.
 .
 This package contains a version of vim compiled with a rather
 standard set of features.  This package does not provide a GUI
 version of Vim.  See the other vim-* packages if you need more
 (or less).
Homepage: http://www.vim.org/
```

Esempio pacchetto installato in /var/log/dpkg.log

```
2014-11-14 18:11:47 status installed python-astroid:all 1.2.1-1
2014-11-14 18:11:47 configure pylint:all 1.3.1-1 <none>
2014-11-14 18:11:47 status unpacked pylint:all 1.3.1-1
2014-11-14 18:11:47 status half-configured pylint:all 1.3.1-1
2014-11-14 18:11:48 status installed pylint:all 1.3.1-1
```

Esempio pacchetto rimosso in /var/log/dpkg.log

```
2014-08-26 17:18:09 startup packages remove
2014-08-26 17:18:09 status installed rsyslog:amd64 5.8.11-3
2014-08-26 17:18:13 remove rsyslog:amd64 5.8.11-3 <none>
2014-08-26 17:18:13 status half-configured rsyslog:amd64 5.8.11-3
2014-08-26 17:18:14 status half-installed rsyslog:amd64 5.8.11-3
2014-08-26 17:18:14 status triggers-pending man-db:amd64 2.6.2-1
2014-08-26 17:18:14 status half-installed rsyslog:amd64 5.8.11-3
2014-08-26 17:18:14 status config-files rsyslog:amd64 5.8.11-3
```

Una volta effettuato un opportuno *parsing* dei file appena descritti, aggiungendo installazioni e rimozioni alla situazione corrente, riusciamo a conoscere il periodo di tempo in cui un dato CPE 2.2 è stato presente nella macchina in analisi.

4.3.2 Acquisizione DSA

Riusciti ad ottenere i pacchetti software da analizzare, si scarica la lista delle vulnerabilità per cui è disponibile un aggiornamento (**DSA/list**), dal *repository* locale o remoto secure-testing. Effettuato il *parsing* si associa ad ogni pacchetto la corrispettiva finestra di vulnerabilità, se presente. Per un esempio di informazione contenuta nel file DSA/list vedere 3.2.3

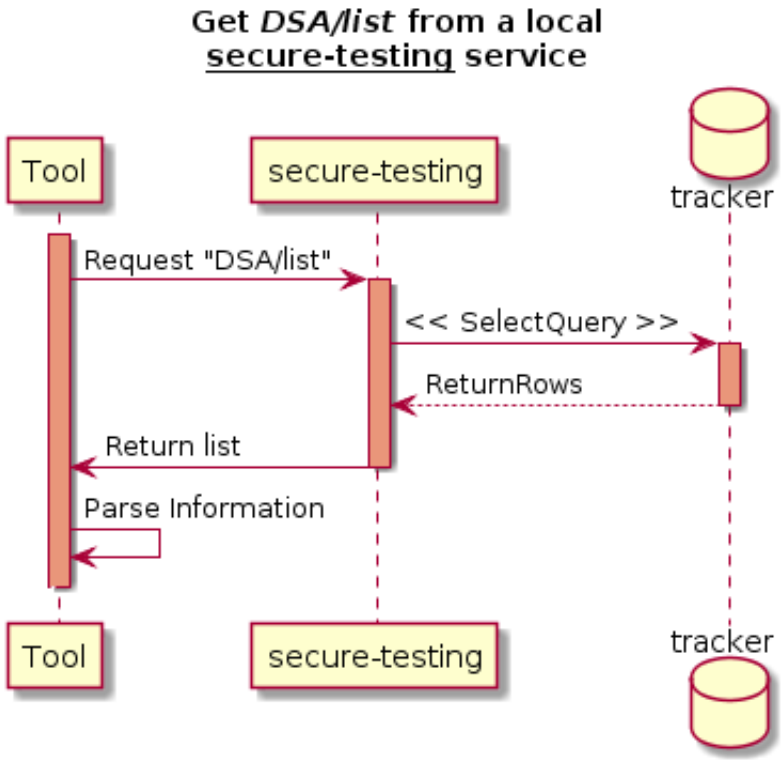

Figura 4.2: Diagramma di sequenza per ottenere la lista dei DSA da un *repository* locale

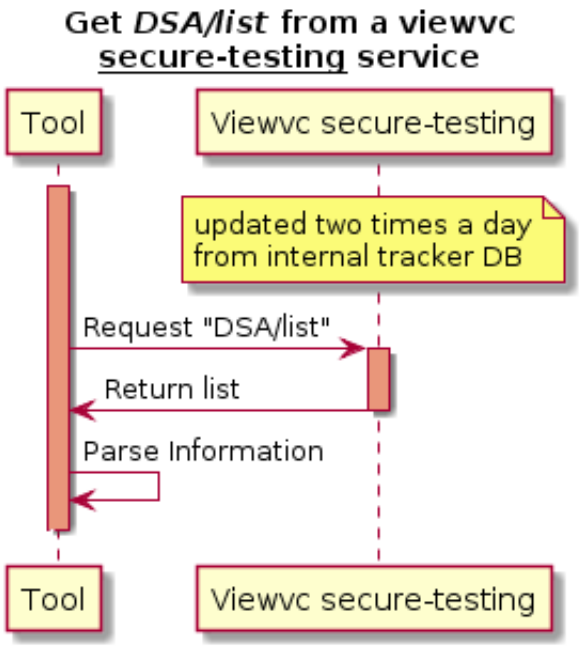

Figura 4.3: Diagramma di sequenza per ottenere la lista dei DSA da un server remoto online

4.3.3 Acquisizione CVE

Ottenere una finestra di vulnerabilità per ogni pacchetto in funzione della data di pubblicazione dei CVE associati, essendo questi ultimi non esclusivi al mondo Debian, risulta più complicato. È necessario infatti scaricare la lista completa ufficiale (disponibile all'indirizzo cve.mitre.org/data/downloads/) contenente per ogni CVE anche la data di rilascio, ed associarla poi alla lista revisionata dal team di sicurezza Debian, con le informazioni indispensabili a sapere se una segnalazione interessa realmente la distribuzione (**CVE/list**).

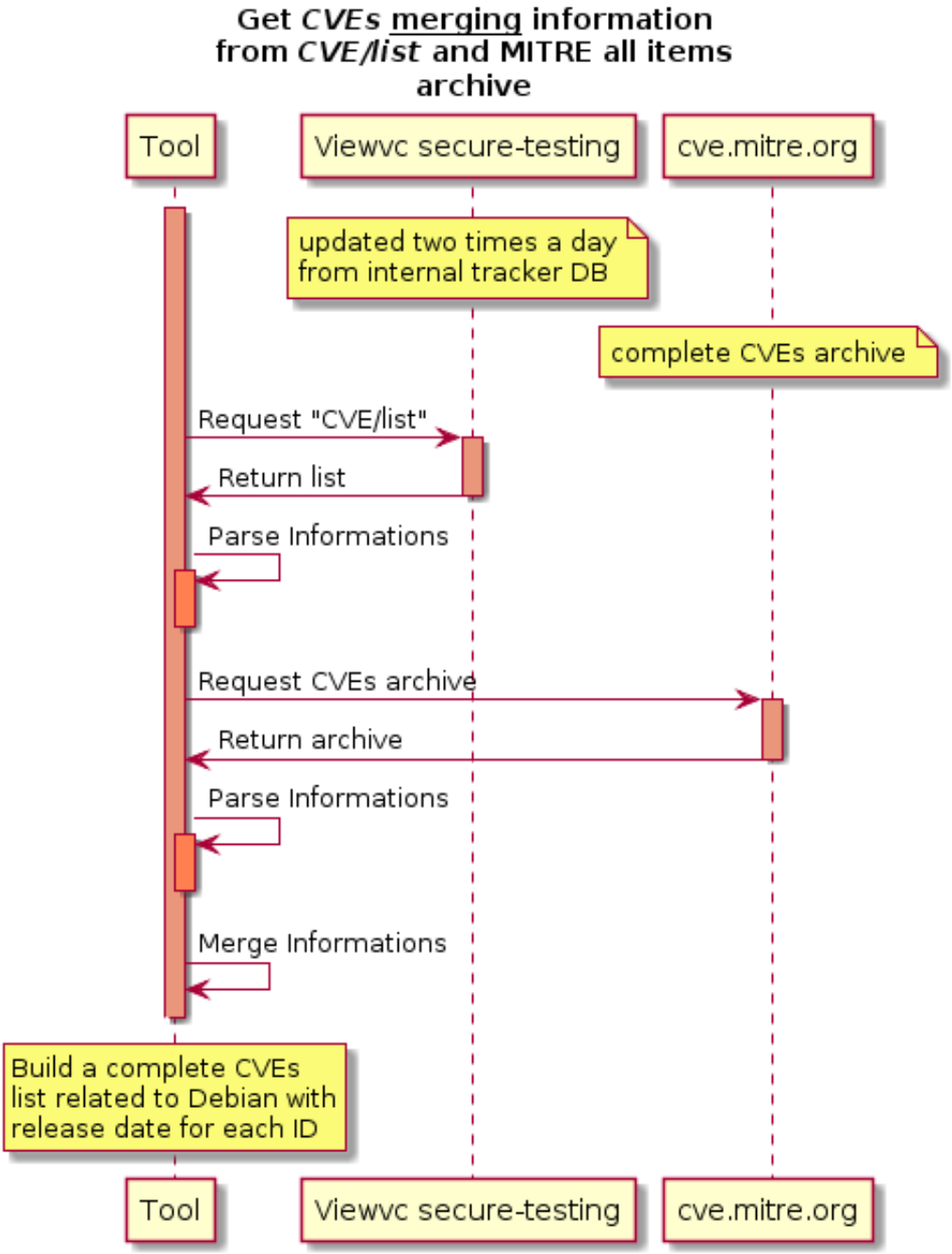

Figura 4.4: Diagramma di sequenza per ottenere la lista dei CVE

4.3.4 Activity Diagram

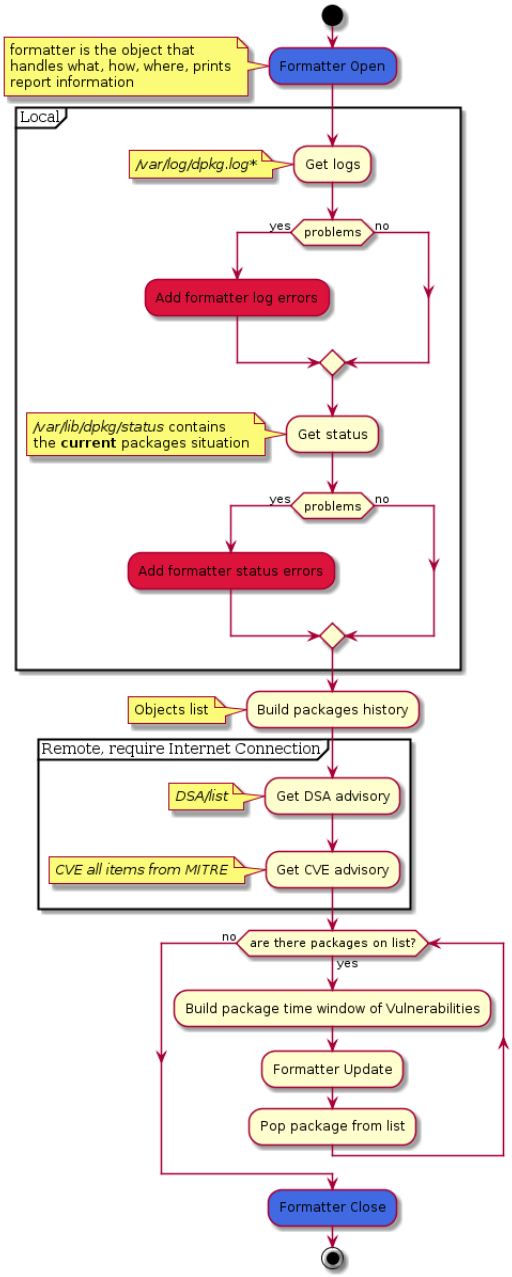

Figura 4.5: Diagramma di attività algoritmo principale

Nel diagramma di attività l'unica parte non spiegata in precedenza riguarda il *formatter*, ossia la parte che si occupa di gestire la formattazione del *report* rilasciato a fine esecuzione. Per mantenere compatibilità con i tipi di formattazione disponibili nel tool debsec, è possibile stampare il risultato dell'analisi su più formati, con diverso grado di dettaglio.

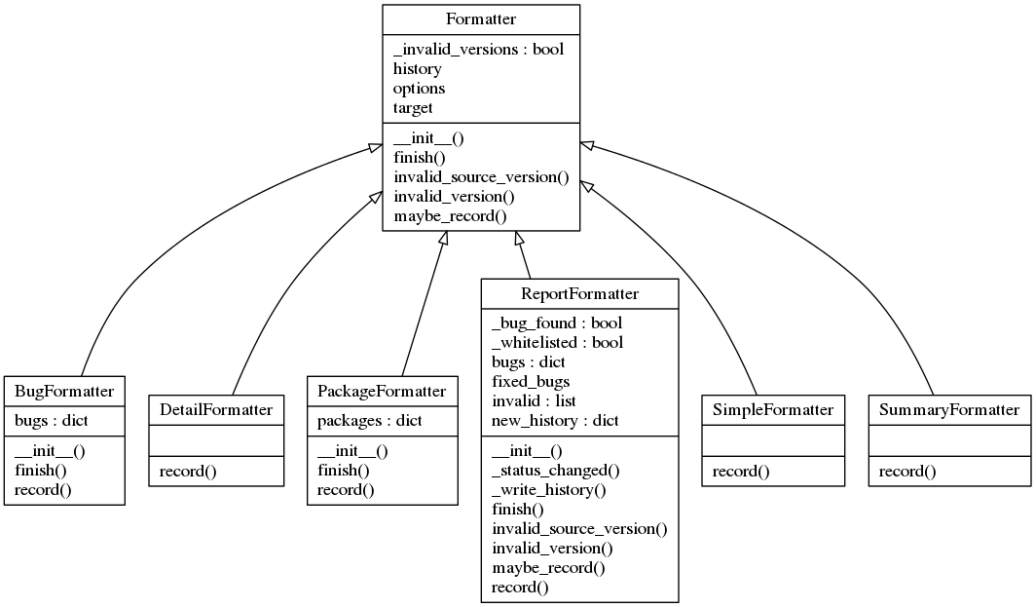

Figura 4.6: Diagramma delle classi che descrive i possibili formati di output

4.4 Limitazioni

Da come si è visto, Lapsus si basa sul contenuto di file che potrebbero venire alterati (*file tampering*) ed è quindi opportuno tenerne conto in fase di analisi. Si consiglia di adottare ulteriori misure di sicurezza, come salvare i file in un *host* remoto non raggiungibile dall'esterno, per esempio su una rete locale. Inoltre, non è ancora prevista una funzionalità che tenga traccia di tutti i software che non appartengono ufficialmente alla distribuzione Debian e/o comunque non sono stati gestiti tramite il tool dpkg.

Conclusioni

Grazie a Lapsus, seppur ancora nella sua prima versione, è possibile analizzare un periodo di tempo passato, al fine di facilitare l'individuazione di problemi di sicurezza nei pacchetti software installati. Al momento non è stato ancora reso possibile astrarlo a tal punto da essere indipendente dalla distribuzione Debian [1] GNU/Linux [2], in quanto le *policy* che riguardano le misure di sicurezza non sono ancora standardizzate ed ogni distribuzione adotta quelle ritenute più opportune alla propria filosofia.
Personalmente ritengo che Lapsus potrebbe essere integrato, all'interno o come modulo, al tool debsecan; sempre di analisi si tratta, con la differenza che, invece di focalizzare l'attenzione sullo stato attuale della macchina *host*, si ricostruisce la sua storia.
La problematica qui affrontata riguarda, dal punto di vista della sicurezza, la gestione di pacchetti software. L'attenzione è esclusivamente focalizzata sulle possibili inadempienze da parte di un amministratore di sistema o degli eventuali ritardi nei rilasci di aggiornamenti.
Per approfondimenti su altri possibili meccanismi di sicurezza adottati da un *package manager*, si consiglia di leggere il seguente articolo [29].

Ringraziamenti

I seguenti ringraziamenti sono rivolti a coloro che mi hanno seguito nel corso di questo progetto, a coloro che mi hanno supportato ed incoraggiato a non mollare, a coloro che ogni giorno lo fanno e mi stanno accanto. (Non sono in ordine di importanza).

Gabriele D'Angelo
Stefano Zacchiroli
Franco Callegati
Marco Ramilli
Jessica Albonetti
Serena Scarpellini
Matteo Francia
Daniele Bellavista

Grazie davvero di cuore,
Giacomo

Lista dei termini

ABI Application Binary Interface. 33

Alioth alioth.debian.org è un server Debian con FusionForge. È molto simile a servizi come sourceforge o savannah ma per gli sviluppatori Debian.. 10, 33

Apache Subversion (SVN) Software versioning and revision control system distribuito come software libero sotto licenza Apache. 10, 33

API Application Programming Interface. 33

CCE Common Configuration Enumaration. 33

CNA CVE Numbering Authority. 4, 33

CPE Common Platform Enumeration. i, 5, 26, 33

CVE Common Vulnerabilities and Exposures. i, 3, 5, 8, 9, 33

CVRF Common Vulnerability Reporting Framework. i, 5, 33

Denial of Service attack (DoS) Attacchi che mirano a negare (*deny*) l'utilizzo o l'accesso a servizi e/o risorse remote, siano esse host o intere reti.. 11, 12, 33

distribuzione Una distribuzione Linux (detta anche distribuzione GNU/Linux o gergalmente distro), è un sistema operativo basato sulla famiglia dei software che utilizzano un kernel Linux, spesso corredato da elementi

del progetto GNU. Tali distribuzioni appartengono della famiglia dei sistemi operativi cosiddetti *Unix–like*.. iii, 33

DSA Debian Security Advisory. 7, 9, 33

Internet Relay Chat (IRC) Protocollo a livello applicazione (Application Layer) che facilita il trasferimento di messaggi in formato testo. Il processo della chat lavora su di un modello di rete client/server. I client IRC sono in grado di comunicare con il server incaricato di inoltrare il messaggio agli altri client. Il suo sviluppo è stato pensato per comunicazioni di gruppo tramite quelli che vengono chiamati *canali*. Può essere anche usato per comunicazioni uno-a-uno tramite messaggi privati e trasferimento dati incluso la condivisione di file.. 8, 33

Intrusion Detection System (IDS) Dispositivi o applicazioni software che controllano la presenza di attività malevoli e violazioni di diritti in un sistema o in una rete riferendo ad una stazione di controllo. Ogni IDS si differenzia per tipologia di approccio all'individuazione di traffico sospetto.. 8, 33

NVD National Vulnerability Database. 33

pacchetto software In informatica è una serie di programmi che si distribuiscono congiuntamente. In senso più specifico, un pacchetto indica un software per computer compresso in un formato archivio per essere installato da un sistema di gestione dei pacchetti o da un programma d'installazione autonomo. http://it.wikipedia.org/wiki/Pacchetto_(software). 33

vulnerabilità Componente (esplicita o implicita) di un sistema, in corrispondenza alla quale le misure di sicurezza sono assenti, ridotte o compromesse, il che rappresenta un punto debole del sistema e consente a un eventuale aggressore di compromettere il livello di sicurezza dell'intero sistema.. iii, 33

Vulnerability Assessment Processo di identificazione, quantificazione e valutazione di vulnerabilità in un sistema. 8, 33

Bibliografia

[1] "Debian – The Universal Operating System." [Online]. Available: https://www.debian.org/

[2] "gnu.org." [Online]. Available: https://www.gnu.org/gnu/linux-and-gnu.html

[3] "The CERT Division | SEI | CMU." [Online]. Available: http://www.cert.org/index.cfm

[4] CERT, "Computer Forensics," pp. 1–5, 2008. [Online]. Available: https://www.us-cert.gov/sites/default/files/publications/forensics.pdf

[5] K. Kent, S. Chevalier, T. Grance, and H. Dang, "Guide to integrating forensic techniques into incident response," *NIST Special Publication*, 2006. [Online]. Available: http://cybersd.com/sec2/800-86Summary.pdf

[6] "The MITRE Corporation." [Online]. Available: http://www.mitre.org/

[7] "CVE - CVE Abstraction Content Decisions: Rationale and Application ." [Online]. Available: http://cve.mitre.org/cve/editorial_policies/cd_abstraction.html

[8] "CVE - Common Vulnerabilities and Exposures (CVE) ." [Online]. Available: http://cve.mitre.org/

[9] "CVE - Terminology ." [Online]. Available: http://cve.mitre.org/about/terminology.html

[10] "Common Platform Enumeration (CPE)." [Online]. Available: http://scap.nist.gov/specifications/cpe/

[11] "CVE - CVE Usage of CVRF ." [Online]. Available: https://cve.mitre.org/cve/cvrf.html

[12] "The Common Vulnerability Reporting Framework (CVRF) | ICASI." [Online]. Available: http://www.icasi.org/cvrf

[13] "SecurityFocus." [Online]. Available: http://www.securityfocus.com/bid

[14] "Debian Social Contract." [Online]. Available: https://www.debian.org/social_contract

[15] "Debian Mailing Lists – Index for debian-security-announce." [Online]. Available: https://lists.debian.org/debian-security-announce/

[16] "Debian – Debian Releases." [Online]. Available: https://www.debian.org/releases/

[17] "Debian – Security Information." [Online]. Available: https://www.debian.org/security/

[18] "Securing Debian Manual." [Online]. Available: https://www.debian.org/doc/manuals/securing-debian-howto/

[19] "CVE - Requirements and Recommendations for CVE Compatibility - Debian Security Advisories ." [Online]. Available: https://cve.mitre.org/compatible/questionnaires/14.html

[20] "Security Bug Tracker." [Online]. Available: https://security-tracker.debian.org/tracker/

[21] "[secure-testing] Index of /." [Online]. Available: https://anonscm.debian.org/viewvc/secure-testing/

[22] "CVE - Download CVE ." [Online]. Available: https://cve.mitre.org/data/downloads/

[23] "Debian bug tracking system." [Online]. Available: https://www.debian.org/Bugs/

[24] "Securing Debian Manual - Security tools in Debian." [Online]. Available: https://www.debian.org/doc/manuals/securing-debian-howto/ch-sec-tools.en.html

[25] "gnu.org." [Online]. Available: http://www.gnu.org/software/wget/

[26] "Overview — python-apt 0.9.3.7 documentation." [Online]. Available: http://apt.alioth.debian.org/python-apt-doc/

[27] M. Glinz, "On non-functional requirements," in *Requirements Engineering Conference, 2007. RE'07. 15th IEEE International*. IEEE, 2007, pp. 21–26.

[28] S. Adolph, A. Cockburn, and P. Bramble, *Patterns for Effective Use Cases*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[29] J. Cappos, J. Samuel, S. Baker, and J. H. Hartman, "A look in the mirror: Attacks on package managers," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 565–574.

[30] "debsecan." [Online]. Available: http://www.enyo.de/fw/software/debsecan/

[31] "Debian – Debian Security Audit Project." [Online]. Available: https://www.debian.org/security/audit/

[32] "Apt - Debian Wiki." [Online]. Available: https://wiki.debian.org/Apt

[33] "Debian testing security team." [Online]. Available: http://testing-security.debian.net/

[34] “Making Security Measurable.” [Online]. Available: http://makingsecuritymeasurable.mitre.org/index.html