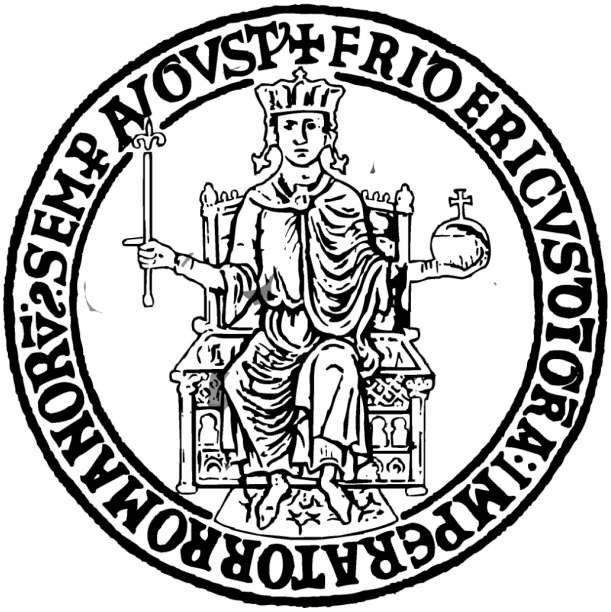
Universita' degli studi di Napoli Federico II



Corso di laurea magistrale in Informatica

Corso di Network Security

Prof. Simon Pietro Romano

Analisi del framework Shodan e del suo utilizzo come strumento di hacking

degli Studenti:

Marco Urbano N97000268

Ciro Brandi N97000269

Anno accademico 2019/2020

Contents

1 Introduzione

> "Se conosci il nemico e te stesso, la tua vittoria è sicura. Se conosci te stesso ma non il nemico, le tue probabilità di vincere e perdere sono uguali. Se non conosci il nemico e nemmeno te stesso, soccomberai in ogni battaglia.
> [1]"
>
> Sun Tzu, L'arte della guerra.

All'epoca di Sun Tzu gli attacchi erano studiati mediante spie che analizzavano il nemico dall'esterno e dall'interno per riuscire a stimare le dimensioni dell'esercito nemico, la dotazione di quest'ultimo e, cosa ancor piu' importante di tutte, quando avrebbe sferrato l'attacco. Oggi, a piu' di duemila anni da quando il trattato militare piu' famoso ha visto la luce, seguiamo ancora minuziosamente i dettami del grande Generale e Filosofo cinese ma il campo di battaglia si e' trasferito anche nel cyberspazio. Difatti ogni attacco che si rispetti prevede una fase di studio del/i target d'interesse per definire in maniera minuziosa tutti i punti deboli di questo/i ultimo/i in modo da incrementare le probabilita' di successo dell'operazione. Quelle che seguono sono identificabili come le prime fasi di una escalation: siamo in sostanza nei panni della spia che anziche' reperire informazioni sull'esercito nemico mira a raccogliere informazioni sulla macchina bersaglio del nostro attacco. L'attaccante eseguira' le seguenti operazioni preliminari:

Footprinting consiste nella raccolta di informazioni sul target per riuscire a tracciare un profilo chiaro e dettagliato delle caratteristiche riguardanti sia la presenza in Internet (i.e. numero di macchine correlate alla stessa entita' di interesse, nomi di dominio, indirizzi IP raggiungibili, meccanismi di controllo degli accessi, architettura della rete di cui il target fa parte, etc.) che inerenti l'entita' del mondo reale che dispone degli host che saranno attaccati (informazioni reperibili mediante l'interrogazione di database pubblicamente disponibili quali *ICANN (Internet Corporation for Assigned Names and Numbers)* o il servizio *WHOIS*, che fino a qualche tempo fa, prima della odierna normativa sulla privacy nota come GDPR, forniva informazioni quali Registrar, Registrant e Registry, utili per

arrivare a tracciare un piccolo identikit dell'identita' di una persona e sfruttabili anche per adottare tecniche di *Social Engineering*)

Scanning e' la fase in cui si analizza il perimetro d'attacco: sondiamo quali servizi sono in "ascolto" per analizzare la reazione della porta a determinati messaggi creati ad arte. In questa fase e' importante utilizzare tecniche che riescano a fornirci quante piu' informazioni possibili sui servizi pronti per essere contattati e, al contempo, garantirci l'anonimato per non mettere in allerta eventuali Intrusion Detection System che potrebbero essere messi a guardia della risorsa da noi richiesta.

Enumeration si conclude l'attivita' di Information Gathering sui servizi in ascolto con questa fase. Quest'ultima attivita' si basa sulla raccolta dell'*informazione aurea* riguardante la versione di ogni servizio disponibile al dialogo presente sul target. E' proprio grazie a questa informazione che e' possibile riflettere sulla miglior strategia per effettuare l'aggressione: il piano d'attacco sara' infatti basato sul numero di *vulnerabilita'* riscontrate sui servizi e sulla combinazione di uno o piu' *exploit* per riuscire a garantirsi l'accesso.

Lo scopo di questo lavoro verte sulla realizzazione di uno strumento per l'automatizzazione delle prime fasi di un attacco. Il servizio grazie al quale e' stato possibile realizzare qualcosa del genere e' il sito web Shodan. Shodan ci permette di richiedere informazioni su particolari indirizzi IP o inserire speciali filtri per ottenere le informazioni che si sarebbero ottenute effettuando le tre fasi precedentemente illustrate e oltre a questo offre servizi Premium per gli utenti registrati che permettono di ottenere informazioni ancora piu' dettagliate a seconda delle caratteristiche del target. Di seguito e' illustrato il funzionamento di ShodanGuru, un semplice script scritto in Python che garantisce l'interfacciamento con Shodan mediante le API e restituisce all'utente le informazioni in un formato testuale facilmente intelligibile. Questo documento termina inoltre con una descrizione delle possibilita' offerte da Shodan e con degli spunti per estendere ShodanGuru al fine di aggiungere nuove potenzialita'.

2 Shodan

Shodan è definito dal suo stesso creatore il *"motore di ricerca più spaventoso al mondo"*. A renderlo cosi terrificante sono proprio gli utenti che utilizzano shodan: gli hacker. A differenza dei classici motori di ricerca come *Google* o *Bing*, Shodan scansiona la rete alla ricerca di nodi di rete che compongono internet, restituendo risultati su server, computer, router, stampanti, webcam, semafori, centrali elettriche e perfino impianti di condizionamento industriali e turbine eoliche. Le scansioni effettuate permettono di classificare tutti i dispositivi e categorizzarli per creare un enorme database che può essere interrogato dagli hacker per recuperare tutte le informazioni necessarie ad un futuro attacco. Il funzionamento di Shodan non si distacca molto da quello dei tradizionali motori di ricerca, ciò che cambia sono solo le chiavi di ricerca. Mentre per *Google o Bing* ci interessiamo di cercare un sito web o immagini, con Shodan si ha la possibilità di ricercare un computer specificando un IP specifico oppure server dotati di un particolare software (come ad esempio Apache) conoscendo qual'è la versione installata sulla macchina e ottenere di conseguenza tutte le vulnerabilità associate al dato servizio con indicazioni sull'exploit da impiegare per l'attacco. Com'è possibile notare tutte queste informazioni sono difficilmente reperibili per i classici motori di ricerca, ma Shodan è in grado di metterle a disposizione di chiunque. Shodan per come è concepito è un arma che nelle mani di professionisti interessati alla sicurezza può risultare uno strumento utile o a dir poco indispensabile, ma la capacità di rendere disponibili a tutti le vulnerabilità di un nodo di rete definisce il lato oscuro di Shodan. Esso fornisce la possibilità di individuare falle di sicurezza dei dispositivi e quindi renderli potenziali target per gli hacker sulla rete. Ancor piu' preoccupante e' che con l'utilizzo di Shodan e con soli pochi clic si è riuscito a controllare un autolavaggio, una pista di pattinaggio e il sistema di controllo centralizzato del traffico negli USA. Tutto questo è avvenuto perchè i router e i server erano privi di ogni sistema di sicurezza[2]. Shodan non deve essere visto solo come strumento malevol: se utilizzato in modo eticamente corretto permette di avere uno strumento capace di segnalare gli innumerevoli bug che affliggono migliaia di nodi connessi in tutto il mondo. In questo modo è possibile conoscere in anticipo le contromisure da mettere in atto affinche gli hacker non possano sfruttare queste vulnerabilità per conquistare i nostri dispositivi di rete.

Attualmente Shodan permette di eseguire ricerche e restituire dieci risultati agli utenti privi di account e cinquanta risultati agli utenti che ne possiedono uno. Uno *Shodan Premium Account* abilita diverse funzionalità tra cui:

1. **Bulk DataFeed**: Download di tutti i dati raccolti da Shodan per creare un proprio database di dispositivi connessi a Internet.
2. **On-Demand Scanning**: pianificare scansioni di reti che vanno dai singoli IP fino all'intera Internet.
3. **Unlimited Access**: accesso illimitato a Shodan.

2.1 Aspetti tecnici

Shodan è un motore di ricerca orientato all IoT, in particolare orientato sui sistemi SCADA. L'unità base che Shodan raccoglie è il *banner*. Il banner è un'informazione testuale che descrive un servizio su un dispositivo che varia notevolmente a seconda della tipologia di servizio. Oltre al banner, Shodan acquisisce anche *metadati* sul dispositivo come la **posizione geografica**, **hostname**, **sistema operativo**, **etc**. Mostriamo un esempio di banner[1] restituito da Shodan con annessi metadati.

```
{
  "timestamp": "2014-01-16T08:37:40.081917",
  "hostnames": [
    "99-46-189-78.lightspeed.tukrga.sbcglobal.net"
  ],
  "org": "AT&T U-verse",
  "guid": "1664007502:75a821e2-7e89-11e3-8080-808080808080",
  "data":
      "NTP\nxxx.xxx.xxx.xxx:7546\n68.94.157.2:123\n68.94.156.17:123",
  "port": 123,
  "isp": "AT&T U-verse",
  "asn": "AS7018",
  "location": {
    "country_code3": "USA",
    "city": "Atlanta",
    "postal_code": "30328",
    "longitude": -84.3972,
    "country_code": "US",
    ...
```

[1] Descrizione del banner al seguente link: https://developer.shodan.io/api/banner-specification

```
  },
  "ip": 1664007502,
  "domains": [
    "sbcglobal.net"
  ],
  "ip_str": "99.46.189.78"
}
```


2.2 Shodan Crawler

I crawler Shodan scandagliano il web 24/7 e aggiornano il database in tempo reale ottenendo cosi l'immagine più recente di Internet. Essi sono presenti nei paesi di tutto il mondo, tra cui: ***USA (East and West Coast), Cina, Islanda, Francia, Taiwan, Vietnam, Romania, Repubblica Ceca.*** I dati vengono raccolti in modo da prevenire distorsioni geografiche dovute ai blocchi di indirizzi IP applicati dagli Stati. La distribuzione dei crawler garantisce che qualsiasi tipo di blocco a livello nazionale non influisca sulla raccolta dei dati. Ogni crawler esegue la seguente sequenza di operazioni:

1. Generare un ***indirizzo IPv4 casuale***.
2. Generare una ***porta casuale***.
3. Controllare l'indirizzo IPv4 casuale sulla porta casuale e prendere un **banner**.
4. ***Ritorna alla prima operazione e ripeti il ciclo***.

Ciò significa che i crawler non eseguono la scansione di intervalli di rete incrementali ma bensì la scansione viene eseguita in modo completamente casuale per garantire una copertura uniforme di Internet e prevenire distorsioni nei dati. I crawler tentano di analizzare il testo del banner principale e di analizzare qualsiasi informazione utile, come l'acquisizione di schermate di servizi desktop remoti o la memorizzazione di un elenco di peer per Bitcoin. Shodan utilizza le seguenti due tecniche avanzate di analisi dei dati:

- **Web Components**
- **Cascading**

2.2.1 Web Components

I crawler cercano di determinare le tecnologie Web utilizzate su un sito Web. L'header HTTP e il codice HTML viene scomposto e analizzato per determinare le componenti del sito web. Le informazioni risultanti sono archiviate nella proprietà *http.components*. La proprietà è un dizionario di tecnologie, in cui la chiave è il nome della tecnologia (ad esempio jQuery) e il valore è un altro dizionario con una proprietà di categorie. La proprietà *categories* è un elenco di categorie associate alla tecnologia.

```
"http": {
  ...
  "components": {
    "jQuery": {
      "categories": ["javascript-frameworks"]
    },
    "Drupal": {
      "categories": ["cms"]
    },
    "PHP": {
      "categories": ["programming-languages"]
    }
  },
  ...
},
```

La proprietà *http.components* indica che il sito Web esegue il sistema CMS Drupal, che a sua volta utilizza jQuery e PHP.

2.2.2 Cascading

Se un banner restituisce informazioni su peer o contiene in altro modo informazioni su un altro indirizzo IP che esegue un servizio, i crawler tentano di eseguire una cattura del banner su quel IP/servizio. Ad esempio: la porta predefinita per la linea principale DHT (Distributed Hash Table - utilizzata da Bittorrent) è 6881. Il banner per tale nodo DHT ha il seguente aspetto:

```
DHT Nodes
97.94.250.250 58431
150.77.37.22 34149
113.181.97.227 63579
252.246.184.180 36408
...
```

il crawler lancia nuove richieste di cattura del banner per tutti i peer. Nell'esempio sopra, il crawler avvierebbe una scansione per IP 97.94.250.250 sulla porta 58431 utilizzando il banner grabber su DHT per IP 150.77.37.22 sulla porta 34149 e così via. Una singola scansione per IP può causare scansioni a cascata se i dati della scansione iniziale contengono informazioni su altri potenziali host.

2.2.3 Servizi in Shodan: SSL

Descriveremo un esempio di servizio, in questo caso SSL per descrivere gli standard che shodan applica quando il banner viene recuperato. I banner per i servizi SSL, come HTTPS, includono non solo il certificato SSL ma anche molto altro. Tutte le informazioni SSL raccolte sono archiviate nella proprietà *ssl* sul banner. L'utilizzo di SSL comporta anche ottenere informazioni sulle vernerabilità della versione SSL utilizzata. Heartbleed è una vulnerabilità nella libreria di crittografia OpenSSL, che è un'implementazione ampiamente usata del protocollo TLS. Se il servizio è vulnerabile a Heartbleed, il banner conterrà due proprietà aggiuntive. *opts.heartbleed* contiene la risposta non elaborata dall'esecuzione del test Heartbleed rispetto al servizio. Il test che i crawler eseguono acquisiscono solo un piccolo overflow dei dati per confermare che il servizio è interessato da Heartbleed ma non acquisisce abbastanza dati per recupeare le chiavi private. I crawler aggiungono nel caso di Heartbleed **CVE-2014-0160** all'elenco *opts.vulns* se il dispositivo è vulnerabile. Tuttavia, se il dispositivo non è vulnerabile, viene aggiunto "**! CVE-2014-0160**". Se una voce in *opts.vulns* è preceduta da un **!** oppure **-** il servizio non è vulnerabile al CVE specificato.

```
"opts": {
  "heartbleed": "... 174.142.92.126:8443 - VULNERABLE\n",
  "vulns": ["CVE-2014-0160"]
}
```

Normalmente, quando un browser si connette a un servizio SSL, negozia la versione SSL che vorrebbe utilizzare con il server. I crawler Shodan iniziano i test SSL eseguendo una normale richiesta, dove negoziano con il server. Tuttavia, in seguito tentano anche esplicitamente di connettersi al server utilizzando una versione SSL specifica. In altre parole, i crawler tentano di connettersi al server utilizzando *SSLv2, SSLV3, TLSv1.0, TLSv1.1 e TLSv1.2* esplicitamente per determinare tutte le versioni supportate dal servizio SSL. Le informazioni raccolte vengono rese disponibili nel campo *ssl.versions*:

```
"versions":["TLSv1","SSLv3","-SSLv2","-TLSv1.1","-TLSv1.2"]
```

Se la versione inizia con un **-**, il dispositivo non supporta quella versione SSL, altrimenti la supporta. Nel campo *ssl.chain* viene indicata la catena di certificati da una Certification Authority all'utente finale

2.3 Shodan: Caratteristiche aggiuntive

2.3.1 Mappatura dei risultati

Shodan mostra visivamente i risultati forniti in modo che siano più facile da leggere rispetto alla forma testuale. Visualizza centinaia di risultati attraverso la funzione Mappa.

2.3.2 Tracciamento degli exploit

Shodan raccoglie vari exploit e vulnerabilità digitali da fonti come Exploit DB, CVE e Metasploit e li fornisce attraverso un'interfaccia di ricerca web. Attraverso le API è possibile recuperare il banner degli exploit dalle diverse fonti precedentemente citate. [2]

2.3.3 Rapporti sullo stato

Per qualsiasi query di ricerca, Shodan permettere di fare un'istantanea di come i risultati della ricerca sono distribuiti geograficamente in quel momento. I rapporti generati da Shodan forniscono una panoramica generale dei risultati mostrati attraverso grafici e diagrammi dettagliati.

2.3.4 Ricerche generate dalla comunità

Ogni query di ricerca eseguita dagli utenti viene salvata per elaborare delle statistiche su quelle più utilizzate o più interessanti o utili. Bisogna quindi tener presente quindi che qualsiasi ricerca eseguita diventa di dominio pubblico.

[2] https://developer.shodan.io/api/exploit-specification

3 ShodanGuru

Abbiamo scelto il nome *"ShodanGuru"*, con una iperbole scherzosa, per indicare che questo script e' capace di restituire, all'utente che ne faccia richiesta, tutte le informazioni su un determinato host o su un insieme di host con determinate caratteristiche: proprio come un *Guru* che dovrebbe avere la risposta a tutti i quesiti che gli vengono posti (in questo caso solo quesiti riguardanti le potenziali vulnerabilita' che l'attaccante vuole sfruttare!). Questo capitolo e' diviso in due sezioni principali: in *"Progettazione di alto livello"* sono riportati tutti i diagrammi UML che descrivono lo script in *maniera statica e dinamica* e indicano i requisiti indicati in fase di analisi tramite il Diagramma dei Casi D'uso, mentre in "Programmazione di basso livello" vengono esaminate le porzioni di codice ritenute piu' interessanti per la comprensione dello script.

3.1 Progettazione di alto livello

3.1.1 Requisiti

Dato che l'obiettivo prefissato di questo lavoro e' *"Analizzare il framework Shodan e il suo utilizzo come strumento di Hacking"*, sono stati inquadrati, come prevede la corretta progettazione secondo i principi di Ingegneria del Software, i seguenti ***requisiti funzionali e non funzionali***.

Requisiti funzionali permettere all'utente di ottenere le informazioni che e' possibile ottenere eseguendo le attivita' di Footprinting, Scanning ed Enumeration riguardo singoli target, di cui si conosce gia' l'indirizzo IP, o un insieme di target che abbia determinate caratteristiche di software, geografiche, etc.

Requisiti non funzionali .

- **Intelligibilita' dei dati raccolti**
- **Estensibilita' e riuso del codice**

Oltre a garantire i requisiti funzionali e' stata posta particolare enfasi sul requisito di Estensibilita' e riuso del codice a causa della continua evoluzione di Shodan. Per garantire questo requisito si e' progettato questo semplice script interamente ad oggetti, cosi' da permettere all'utilizzatore di personalizzare ShodanGuru a proprio piacimento. Il requisito di intelligibilita' pone invece particolare enfasi sul mostrare i dati di maggiore interesse: questo requisito e' stato rispettato permettendo diverse modalita' di visualizzazione dei dati.

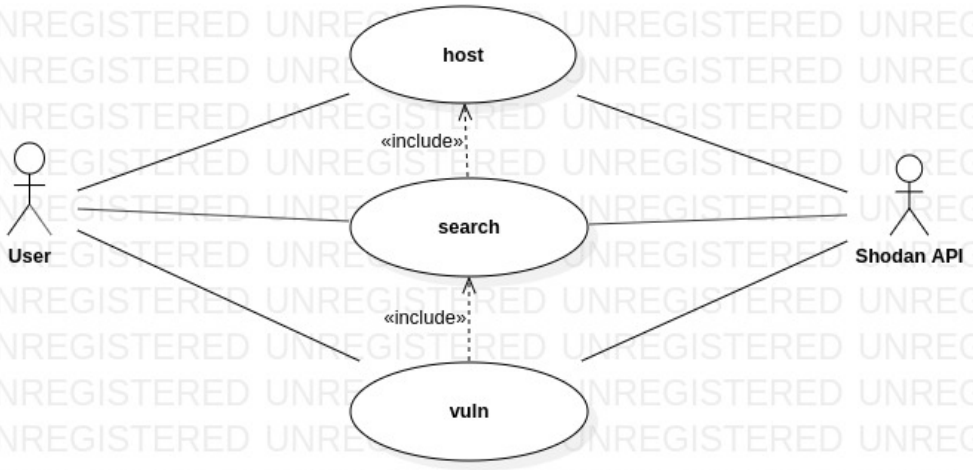
3.1.2 Use case diagram

Al fine di garantire l'unico requisito funzionale individuato sono state individuate tre funzionalita'. L'utente puo' scegliere tra le seguenti attivita':

Host l'utente immette l'indirizzo ip e specifica il livello di *verbosita'* dei dati da mostrare. Se quest'ultimo viene omesso, sara' mostrata l'informazione essenziale, senza mostrare i dettagli di ogni vulnerabilita', bensi' solo i codici identificativi da cui ricavare i dettagli sull'exploit.(modalita' di default).

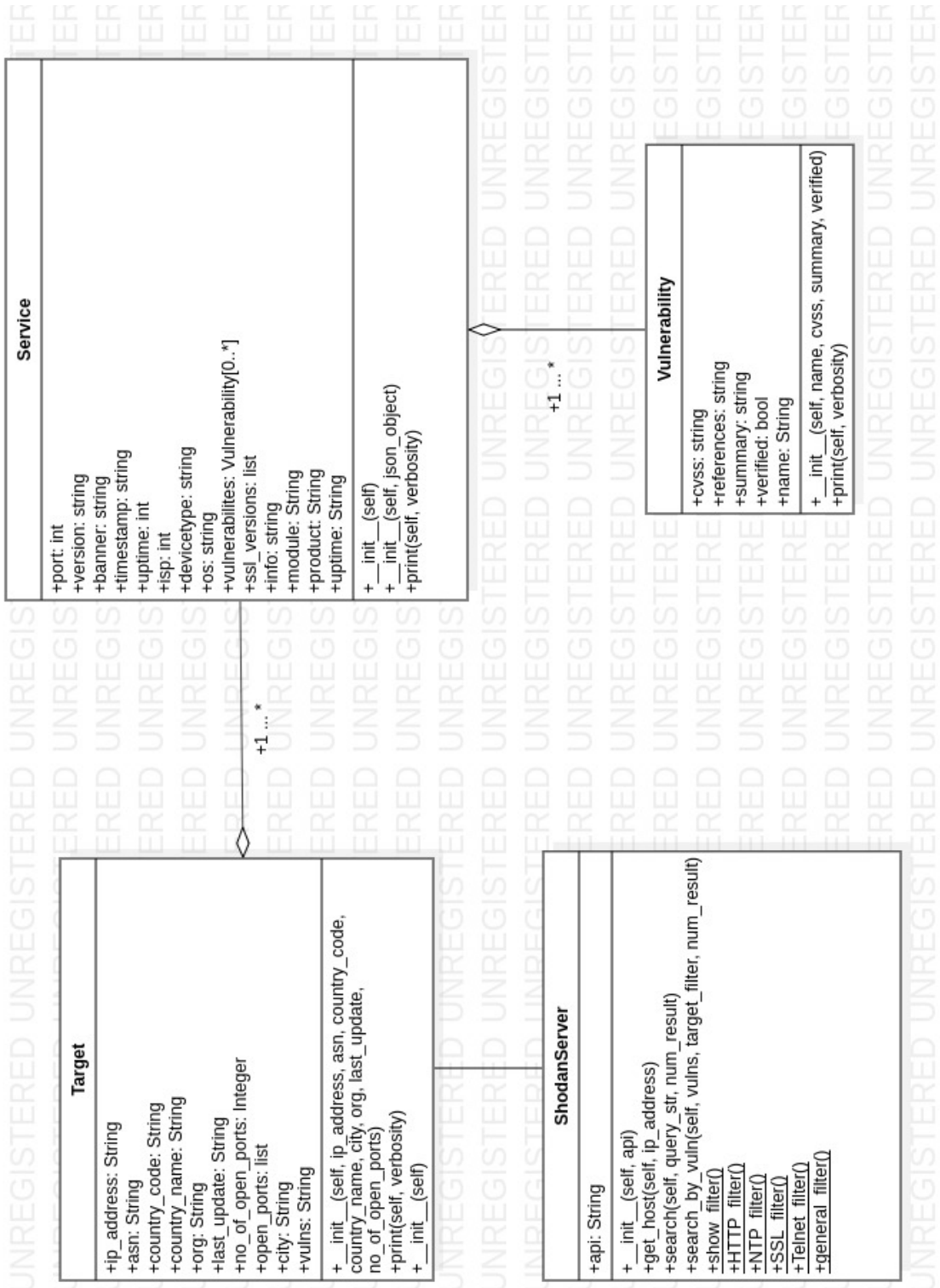
Search l'utente specifica un filtro d'interesse per recuperare informazioni solo sugli host che ricadono nella categoria specificata (i.e. **city:Napoli**, per richiedere che siano recuperati host con indirizzo proveniente da Napoli , o **product:Apache**, per recuperare informazioni sugli host che ospitano un Apache Web Server) ed un intero per limitare il numero dei risultati da produrre.[3] Anche in questo caso si puo' specificare o meno il livello di *verbosita'*.

Vuln in questa speciale modalita' l'utente inserisce un filtro, una lista di codici di vulnerabilita' e un intero per limitare il numero di risultati restituiti per recuperare le informazioni di tutti gli host che soffrono di almeno una delle vulnerabilita' indicate. La verbosita' e' quella di default.

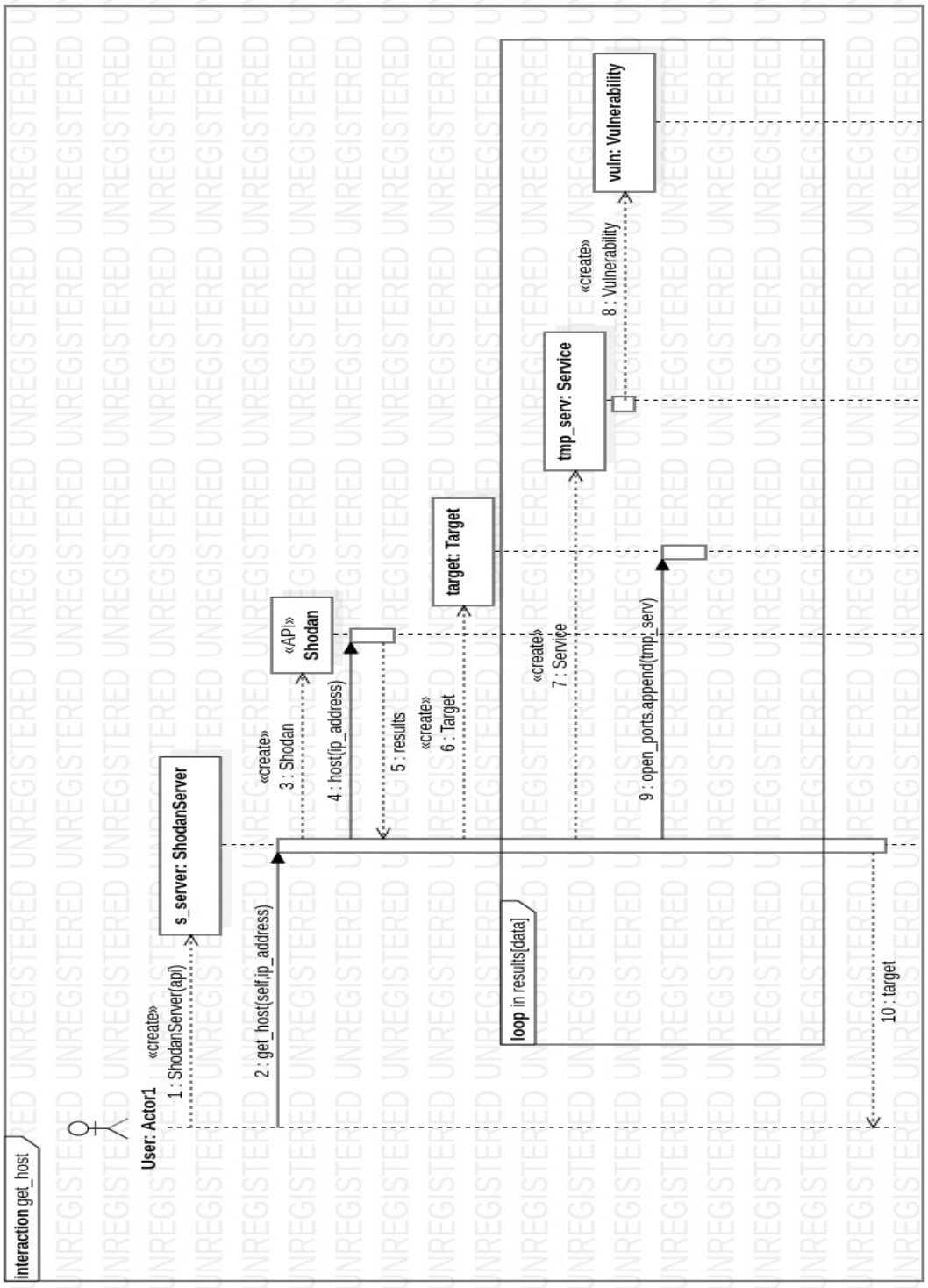


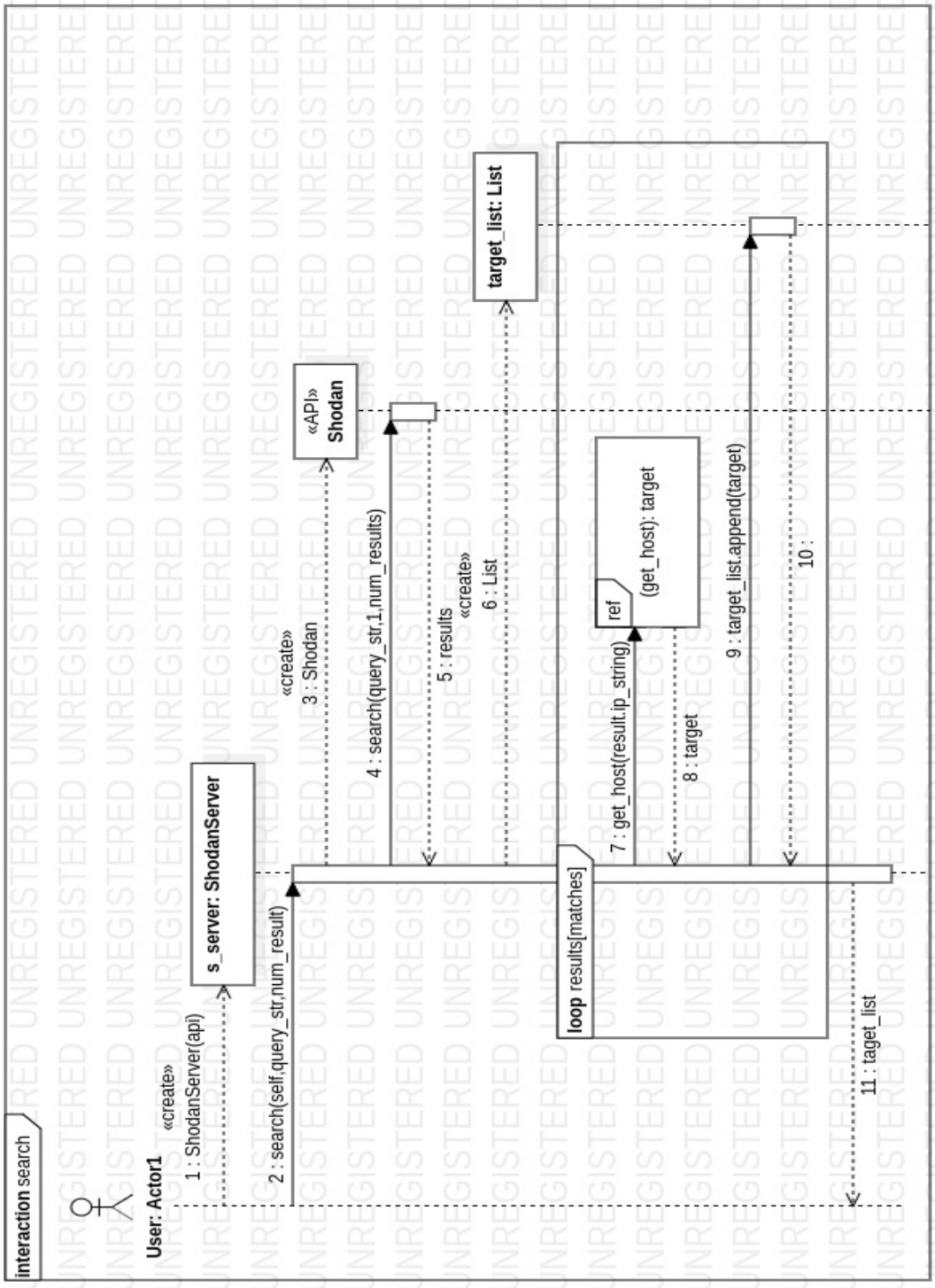
[3]Si ricorda che solo gli utenti premium possono effettuare query con filtri attraverso le API. Inoltre il numero di search query e' limitato e viene rinnovato mensilmente per i membri Shodan.

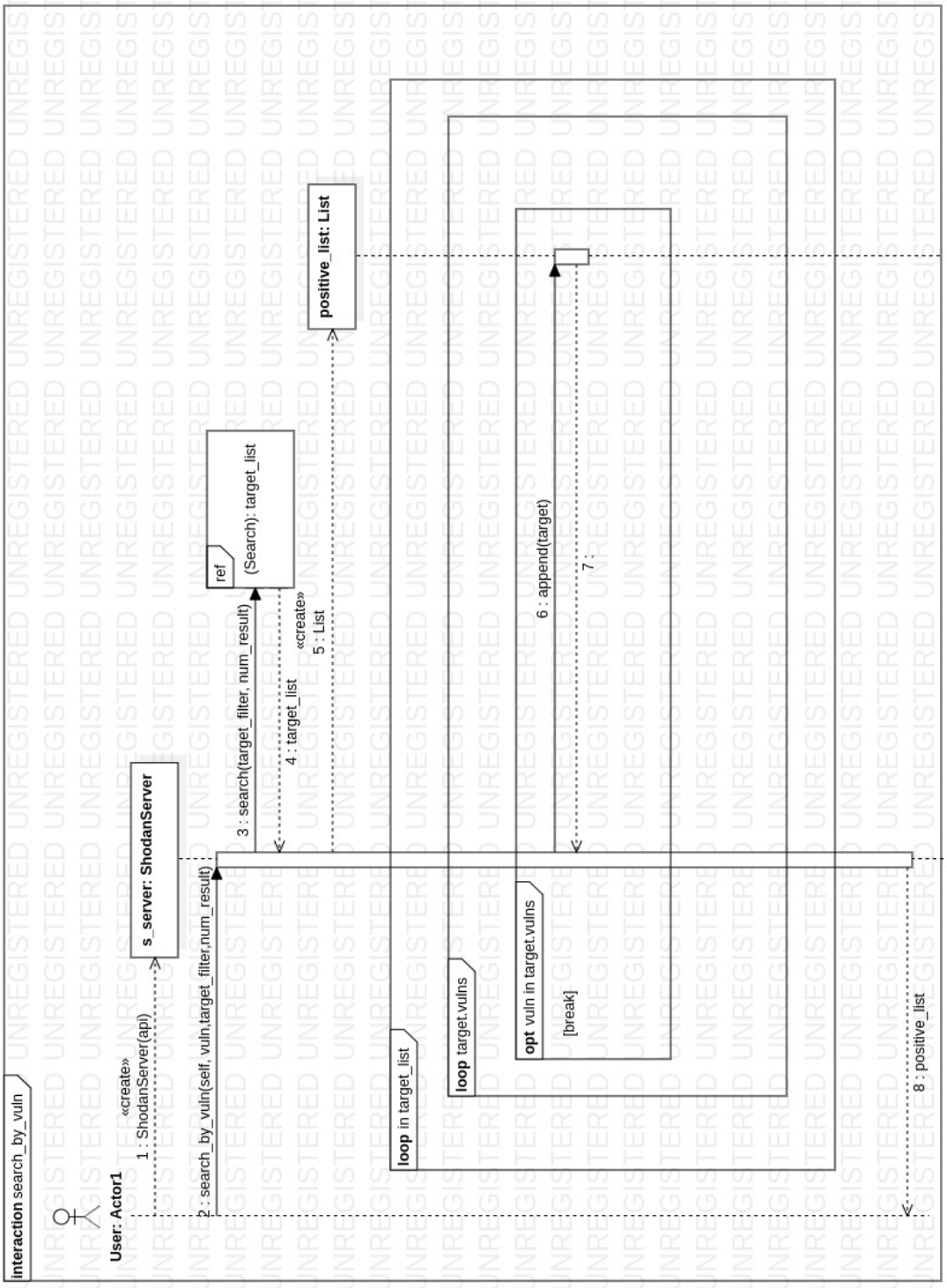
3.1.3 Class diagram



3.1.4 Sequence diagram




interaction search
User: Actor1
«create»
1 : ShodanServer(api)
s_server: ShodanServer
2 : search(self,query_str,num_result)
«create»
3 : Shodan
«API»
Shodan
4 : search(query_str,1,num_results)
5 : results
«create»
6 : List
target_list: List
loop results[matches]
ref
(get_host): target
7 : get_host(result.ip_string)
8 : target
9 : target_list.append(target)
10 :
11 : taget_list


interaction search_by_vuln
User: Actor1
«create»
1 : ShodanServer(api)
s_server: ShodanServer
2 : search_by_vuln(self, vuln,target_filter,num_result)
3 : search(target_filter, num_result)
ref
(Search): target_list
4 : target_list
«create»
5 : List
positive_list: List
loop in target_list
loop target.vulns
opt vuln in target.vulns
[break]
6 : append(target)
7 :
8 : positive_list

3.2 Programmazione di basso livello

3.2.1 Tecnolgie utilizzate

Le tecnologie utilizzate per l'implementazione di ***ShodanGuru*** sono: ***Python***, ***IDE Pycharm*** e ***API Rest Shodan***. Le caratteristiche principali che hanno fatto ricadere la scelta su Python sono: Portabilità e Pulizia del codice, mentre la scelta delle API Shodan è stata vincolata dal plan associato all'account di sviluppo, ovvero come detto in precedenza da un account standard (account registrato).

3.2.2 Analisi del codice

Le API Shodan si dividono in ***API Rest*** e ***API Stream***[4]: dove le prime forniscono metodi per ottenere le informazioni che sono state gia' raccolte dai crawler dai database di Shodan, mentre le seconde restituiscono un feed in tempo reale dei dati che i crawler stanno raccogliendo. ShodanGuru fa uso solo delle prime in quanto e' stato pensato per restituire solo le informazioni che l'utente richiede e non tutta la mole di dati che viene scandagliata dai crawler[5] (che potrebbe risultare di poco interesse). La prima operazione da compiere per ottenere le informazioni da Shodan e' quella di richiamare il costruttore fornito dalle API di Shodan con la rispettiva API Key associata al proprio account.

```
1    def __init__(self, api):
2        self.api = shodan.Shodan(api)
```

Dopo aver ottenuto l'oggetto *api* si possono utilizzare i metodi definiti da Shodan per scegliere come interrogare il database. Per scelta progettuale si e' deciso di rendere *api* un attributo della classe ShodanServer: questa classe fa da intermediario tra Shodan e l'utente che richiede informazioni. Piu' nel dettaglio si occupa di raccogliere i dati in formato ***JSON*** provenienti da Shodan e di istanziare oggetti della classe Target. Cosi' come descritto dal Diagramma dei Casi D'uso, ShodanServer fornisce tre possibilita', tre metodi, all'utente. Andremo ad illustrarle con un approccio Bottom-Up, iniziando dalla piu' semplice, *get_host*, che permette di restituire le informazioni fornendo un singolo Indirizzo IP, fino a *search_by_vuln*, che prende in input un insieme di vulnerabilita' assieme ad un filtro per restituire il numero degli host che soffrono di almeno una delle vulnerabilita' specificate. get_host non fa altro che

[4] https://developer.shodan.io/api

[5] Dato che il codice e' rilasciato sotto licenza GNU General Public License puo' sempre essere esteso per sfruttare anche le API Stream in un secondo momento.

invocare il metodo ***api.host*** fornendo l'indirizzo ip. Dopo questa operazione processa i dati e restituisce un Target.

```
results = self.api.host(ip_address)
    print("Host found on Shodan Database...")

    target = T.Target(results["ip_str"], results["asn"],
                   results["country_code"], results["country_name"],
                   results["city"], results["org"],
                   results["last_update"], len(results["ports"]))

    if "vulns" in results:
      for vuln_code in results["vulns"]:
          target.vulns.append(vuln_code)

    # Checking open ports and initializating services.
    if "data" in results:
        for i in range(0, len(results['data'])):
            # Inizialization service objects
            tmp_serv = S.Service(results['data'][i])
            target.open_ports.append(tmp_serv)

    # Order list by port number
    target.open_ports = sorted(target.open_ports, key=lambda service:
        service.port)
    # Return object
    return target
```

Il metodo *search* invece restituisce un insieme di host che rispettano il filtro inserito dall'utente: ogni host trovato viene inizializzato chiamando il metodo *get_host* appena illustrato.

```
results = self.api.search(query_str, 1, num_result)
for result in results['matches']:
  target_list.append(self.get_host(result["ip_str"]))

return target_list
```

Infine il metodo *search_by_vuln* mette assieme *search* e *get_host* per ottenere informazioni dettagliate riguardo host vulnerabili a determinati exploit.

```
target_list = self.search(target_filter, num_result)
positive_list = []

for target in target_list:
```

```
  for vuln in vulns:
      if vuln in target.vulns:
        positive_list.append(target)

return positive_list
```

La copia di target nella lista positive_list non e' onerosa in termini computazionali dato che si tratta di una copia per riferimento, o anche *shallow copy*[3].

3.2.3 Esempi di utilizzo

Dopo aver inserito la propria API semplicemente copiandola ed incollandola nel main di ShodanGuru, ShodanGuru.py, e' possibile iniziare le ricerche. Di seguito sono presenti alcuni esempi di invocazione per facilitare la comprensione.

```
# Restituisce informazioni con verbosita' di default per l'indirizzo
    193.206.130.6.
python3 ShodanGuru.py host 193.206.130.6
```

```
# Restituisce informazioni con verbosita' alta l'indirizzo
    193.206.130.6. Questa modalita' restituisce tutte le informazioni
    per ogni vulnerabilita'.
python3 ShodanGuru.py host 193.206.130.6 1
```

```
# Restituisce informazioni sui primi 10 host restituiti da Shodan che
    hanno city=Napoli. L'ultimo parametro indica verbosita' bassa,
    ovvero quella di default.
python3 ShodanGuru.py search city:Napoli 10 0
```

```
# Dopo aver reperito i primi 2 host con city=Turin verifica per ogni
    host se soffre di almeno una delle vulnerabilita' specificate. Se
    nessuno degli host reperiti soffre di almeno una vulnerabilita'
    sara' restituito un messaggio di avviso.
python3 ShodanGuru.py vuln CVE-2019-0220 CVE-2018-8011 CVE-2012-4558
    city:Turin 2
```

4 Bibliografia

References

[1] L'arte della guerra. Sun Tzu. VI-V secolo a.C.

[2] https://www.ma-no.org/en/security/shodan-the-scariest-search-engine-on-the-internet

[3] https://medium.com/@thawsitt/assignment-vs-shallow-copy-vs-deep-copy-in-python-f70c2f0ebd86