Journal of Universal Computer Science, vol. 23, no. 4 (2017), 404-430
submitted: 2/11/16, accepted: 27/3/17, appeared: 28/4/17 © J.UCS

Contactless Vulnerability Analysis using Google and Shodan

Kai Simon
(Kai Simon – Consulting, Kaiserslautern, Germany
kai.simon@kaisimon-consulting.de)

Cornelius Moucha
(Kai Simon – Consulting, Kaiserslautern, Germany
cornelius.moucha@kaisimon-consulting.de)

Joerg Keller
(FernUniversität in Hagen, Germany
joerg.keller@fernuni-hagen.de)

Abstract: The increasing number of attacks on internet-based systems calls for security measures on behalf those systems' operators. Beside classical methods and tools for penetration testing, there exist additional approaches using publicly available search engines. We present an alternative approach using contactless vulnerability analysis with both classical and subject-specific search engines. Based on an extension and combination of their functionality, this approach provides a method for obtaining promising results for audits of IT systems, both quantitatively and qualitatively. We evaluate our approach and confirm its suitability for a timely determination of vulnerabilities in large-scale networks. In addition, the approach can also be used to perform vulnerability analyses of network areas or domains in unclear legal situations.

Key Words: Vulnerability analysis, contactless test technique, Shodan, Google
Category: C.2.2, D.4.6, K.6.5

1 Introduction

More and more services are offered publicly available on the Internet. Additionally, larger companies usually employ distributed networks and services for their employees, both internally and externally accessible. At the same time, the software that implements this services becomes more and more complex and harder to secure. This naturally attracts the attention of attackers. In their analysis of the threat landscape, the European Union Agency For Network and Information Security (ENISA) confirmed that web based attacks as well as web application attacks are among the the three most important threats of the year 2015 [ENISA 2016]. Often there are direct consequences of these attacks such as losses in sales, but attacks may also entail indirect and long-term impacts such as reputation loss. Therefore, the demand and interest of providers, system administrators and IT personnel in the security of their systems has increased. However, large-scale audits with conventional penetration tests using mainstream tools such as Nmap or Nessus are usually expensive and time consuming.

Furthermore, legal aspects have to be considered: conventional penetration testing directly contacts the target systems. Particularly in the European Union unsolicited system access is prohibited without the explicit consent of the target system provider as stated in the Directive 2013/40/EU [European Union 2013] of the European Parliament and the Council, Article 2 to 7. This legal constraint is a huge problem for organizations hosting third party services or not having a contractual audit approval. Similarly, in the US, the Computer Fraud and Abuse Act (CFAA) can be used, which primarily aims at commissioning of a criminal offense and not for the operation or possession of potential tools, that can be used for attack preparation.

We present an alternative approach for vulnerability analysis using methods and tools that were originally not invented for this purpose. Instead of manually testing the target systems, we use already existing search engines. This includes general-purpose search engines such as Google or Bing, but also subject-specific alternatives such as Shodan. Currently, the latter are primarily used in the underground [Imperva 2011] [John et al. 2010] or by specialized government authorities [U.S. DHS 2012], but their maturity is inadequate for public or corporate security auditors. These alternative approaches are usually considered only in isolation, because the query signature differs for different target or result purposes which prevents potential synergy effects. Furthermore, the involved technique of data collection as well as the quality and coverage of the data base is still vague.

Using existing work in the field of vulnerability analysis with search engines, we demonstrate that that neither generic nor subject-specific search engines reveal enough data in terms of quantity or quality. We propose a new approach based on refining search terms, combining results from both kinds of search engines and augmenting the results by pairing them with data from publicly available vulnerability databases. The quality and quantity of the vulnerability scan results are evaluated and the results demonstrate that even in comparison with contact-based analysis, i.e. using Nessus, our approach performs well both in terms of precision and recall, is much faster and less stressful for the network environment.

The remainder of this article is structured as follows. In Section 2, we summarize background information and in Section 3 we review related work. Our approach to combine several search engines for contactless vulnerability analysis is presented in Section 4 and evaluated in Section 5. Finally Section 6 provides our conclusions and an outlook on future work.

2 Background

There is a continuously increasing number of attacks on publicly available systems in the past few years [PWC 2016]. Accordingly, there is a growing demand for security audits of IT systems, both corporate internal and by external service contractors. For this purpose, primarily classical tools such as Nmap[1] or Nessus[2] are used. These tools share the common technique of directly contacting the target system (illustrated in Figure 1). Depending on the test configuration, the tests passively scan for existing vulnerabilities or actively try to exploit them. Information on potential vulnerabilities is provided by plugins. These plugins are partly community maintained or commercially sold and updated by the tool provider. Usually, the result report of recognized services and vulnerabilities is in text notation or provided as a web page and therefore requires a manual interpretation and a subsequent risk assessment. Especially the historic course of systems under surveillance according to security criteria involves manual inspections and is challenging.

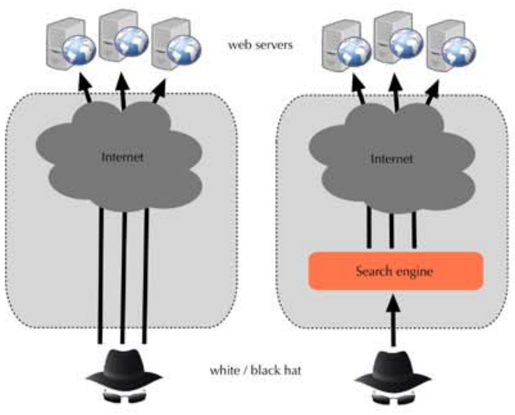

Figure 1: Comparison of contact-based and contactless test methods

The tools and services mentioned directly contact target systems to scan for vulnerabilities. These days, there exist also indirect test techniques. Mainly, they are based on new or already existing search engines, as illustrated on the right hand side in Figure 1. In preparation for eventual user (i.e., any Internet user) search queries, existing Internet websites are accessed in advance and the obtained information is processed and indexed. For providing an up to date level of information, collected data is updated periodically. Search engines such as

[1] `https://nmap.org`
[2] `http://www.tenable.com`

Google or Bing not only process the core content of the websites, but additionally consider meta information such as deployed software and its versions. Based on specifically crafted queries for a search engine, one can obtain information about a target system without directly contacting it. Using the contactless test technique, auditors as well as attackers stay on the sideline and cannot be detected by potential countermeasures of the target system and its infrastructure. Only the examining search engines are visible at the target site. But as search engines repeatedly contact websites for indexing and updating their information, they are usually considered trustworthy. Furthermore, besides general-purpose search engines there also exist so-called subject-specific alternatives. Instead of indexing the main content of the websites, these search engines specifically process the retrieved meta information about systems, e.g. deployed software and their versions. Hence, they provide an interesting opportunity for security auditors as well as attackers to collect data without revealing their identity.

Penetration testing is much broader than retrieving a list of potential vulnerabilities, and requires steps before (e.g. planning) and after (e.g. manual or automated post-processing). But these steps are the same for both, contact-based vulnerability-tools and the contactless method. We compare the results of these two approaches: Both are affected similarly, because they detect potential vulnerabilities, which does not influence our comparison in Chapter 5.

In the following, general-purpose search engines as well as subject area focused alternatives are presented and evaluated for the purpose of vulnerability analysis. First, they are evaluated separately, next, in combination with each other. Finally the quality and quantity of the search results is measured, and potential optimization opportunities are presented.

2.1 General-purpose search engines

According to [de Kunder 2016], information on the Internet consists of more than 45 billion web pages. Finding relevant web pages and information in general is often not trivial. To improve traceability of information and usability for users, the contents of individual websites are systematically and automatically indexed and structured. This task is performed by general-purpose search engines such as Google or Bing. With their help, users can easily search for information using established Internet browsers or special service interfaces.

John et al. [John et al. 2010] discovered that every day specifically specified automated queries are sent to search engines for vulnerability detection. Imperva Inc. [Imperva 2011] has investigated a botnet in 2011 and discovered that an average of 22,000 up to a maximum of 80,000 of these queries were sent to a known search engine whose name is not mentioned. The botnet computers mainly came from Iran, Hungary and Germany. The campaign focused on identifying Cross-site scripting (XSS) vulnerabilities, SQL (Structured Query Language) injection, and outdated software, especially of content management systems (CMSs).

Such a query, called Google dork, is a normal or extended search query, which returns sensitive information or hints of vulnerabilities. Using dork queries in order to discover security vulnerabilities is also called Google dorking or Google hacking. This method was introduced by Long [Long et al. 2007], who collected these dorks on his website. A dork is often composed of two parts: a first part that detects a vulnerability and a second part that is used to focus the target. For example, the following dork looks for obsolete Apache HTTP web servers under the domain "destination.com"[3].

```
Apache/2.0.63 site:destination.com        (1)
```

The suitability and consecutively the appropriateness of results highly depends on the selection of a proper search engine. Although established and well known engines such as Google, Bing or Yahoo are intended to serve the same purpose, their coverage for particular topics differ, both in quantity and quality of the results. Beside the index database, regional orientation plays an important role. The world-wide dominance of the Google search engine is more than 90 percent market share[4]. If the search process was focused on Russia or China, the decision which search engine to use would change because the search engine Yandex has a market penetration of about 40 percent in Russia, whereas in China Baidu is the market leader with about 70 percent market share.

2.2 Subject-specific search engines

In contrast to general-purpose search engines, subject-specific search engines scan the Internet specifically in a defined subject area, such as hosted services, SSL/TLS vulnerabilities or concrete vulnerabilities in Internet-enabled software, such as XSS or SQL injection. Similar to general-purpose search engines, the obtained information is internally processed and aggregated to provide users a fast and comprehensive response for their queries. The major difference of these subject-specific search engines compared to conventional vulnerability analysis tools is the missing direct contact of the users to the target systems, because the information can be directly retrieved from the search engine. Below, some existing search engines for specific subject areas are briefly characterized: Shodan[5], ERIPP (Every Routable IP Project)[6], PunkSPIDER[7] and Netcraft[8].

[3] More dork examples are available on http://www.hackersforcharity.org/ghdb/
[4] `http://gs.statcounter.com`
[5] `https://www.shodan.io`
[6] `http://beta.eripp.com`
[7] `https://www.punkspider.org`
[8] `https://www.netcraft.com`

In summary, ERIPP is not available anymore, and samples of Netcraft and PunkSPIDER indicate only a small set of or at least partly outdated vulnerabilities and indexed websites. Therefore, these search engines will not be considered in the remainder, but only Shodan is chosen for further consideration.

Shodan systematically contacts IP addresses from any region. According to available results, a predefined list of ports is scanned this way. In case of a successful connection to the target system, the retrieved meta-information about running services, so-called "banner information", is stored. As an example, the banner information for an OpenSSH service is `SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u3`. Further information about these meta-information and the processing is given in Section 4. Additionally, publicly available information, such as Fully Qualified Domain Names (FQDNs), complements the entry of an IP address. Shodan is available since 2009. It was developed by John Matherly. According to CNN Money [Goldman 2013], the data base of Shodan is estimated to contain 500 million hosts and their respective IP addresses.

In contrast to general-purpose search engines such as Google, Shodan focuses explicitly on vulnerabilities. Vulnerability detection with Shodan is supported in two ways: On the one hand, requests for specific vulnerabilities can be made. The so-called Shodan queries are comparable to Google dorks. On the other hand, Shodan directly determines selected vulnerabilities and returns them together with the actual query result. The following Shodan query can be used, for example, to detect voice over IP telephones from the manufacturer Snom in network area 11.11.11.0/24 operating on port 5060.

```
port:5060 snom net:11.11.11.0/24                (2)
```

To detect vulnerabilities with this approach, a comprehensive list of high quality Shodan queries is required.

3 Related Work

The usage of specially crafted queries for classic search engines with the intention to collect vulnerability information, so-called "Dorks", was introduced by Johnny Long in [Long et al. 2007] as dork analysis. The term originates from the artificial term "googledork", describing people who introduce vulnerabilities in their systems by misconfiguration. In [Long et al. 2007], primarily the practical execution of dork analysis is presented, without emphasizing the theoretical background. In the meantime, the presented signature database "Google Hacking Database"[9] is outdated, and the quality of dorks is rather weak. In [Zhang et al. 2015], Zhang et al. describe their work on the quantitative evaluation of Google dorks. Their evaluation carried out is primarily concerned with

[9] `https://www.exploit-db.com/google-hacking-database/`

the identifiable vulnerability types, their distribution and potential countermeasures. The method they applied is not reproducible, because necessary raw data is no longer available. Another application of Google dorks is presented by Dalek et al. in [Dalek et al. 2013]. Using dork analysis, they discovered components of censorship (url filters). Several authors discovered the widespread and daily use of these dork analysis techniques, predominately by botnets in the underground [Imperva 2011] [John et al. 2010]. Other publications, such as for example [Billing et al. 2008], [McGuffee and Hanebutte 2013] and [Toffalin et al. 2016] also consider other aspects of dork analysis, but those are of minor relevance for this work.

Shodan, a subject-specific search engine, was used by Radvanovsky and Brodsky in the SHINE project (SHodan INtelligence Extraction) [U.S. DHS 2012]. The purpose of SHINE was the investigation of vulnerabilities in industrial control systems (ICS). According to [U.S. DHS 2012], 7,200 vulnerable systems were discovered in the United States of America. Unfortunately, neither the applied method nor qualitative results were presented.

In his thesis [Schmidt 2015], Schmidt optimized the detection rate of vulnerabilities based on Shodan raw data. His basic approach is to extract identification information from Shodan banner information and to match this information with existing vulnerability databases. The same approach is also used by ShoVAT (Shodan-based vulnerability assessment tool), developed by Genge and Enăchescu [Genge and Enăchescu 2015]. However, their primary focus is on runtime performance optimization and less on qualitative aspects. For qualitative verification, only 40 university addresses were used as a reference set. In addition, only an unreproducible number of Nessus results was used in their comparison. Moreover, banner information retrieved from their test servers and routers seems to be beyond the default configuration of those devices with respect to vulnerability information, which significantly facilitates vulnerability detection. In contrast to such experimentation under laboratory conditions, our approach is to examine whether these alternative methods are applicable for computer security experts (White-hats). Therefore real and unspecified addresses and unprepared systems have to be considered in the quality assessment.

Éireann [Eireann 2011] used Shodan to create a global overview of vulnerable ICS systems. In his study, 7,500 vulnerable systems were identified within an observation period of two years, which have been visualized with respect to their geolocation. In another project using Shodan [Erven and Collao 2015], Erven and Collao discovered medical devices that were only inadequately protected and therefore could be attacked easily. An experiment by Bodenheim [Bodenheim 2014] with honeypots focused on the question of currency and completeness of data that is available in Shodan. In his study, these honeypots were indexed by Shodan within 19 days.

In the last few years, Shodan was mentioned in press reports, whereby a trend towards economic goals emerged. We used the related work in this area as starting basis where it was possible, but pursue a different focus with our work.

Further related work in this area was done by the theses of Scherer [Scherer 2008], Opp [Opp 2014], Oswald [Oswald 2015], Schmidt [Schmidt 2015], von Thaden [von Thaden 2015], Kohl [Kohl 2016], and Sedlmeier [Sedlmeier 2016] under our supervision. Their work evaluates the usage of dorks using the Google search engine as well as optimization approaches for the extraction of identification information from results retrieved with Shodan.

In [Simon 2016] an early stage of the present work was published. Innovations in this paper are the optimization of both approaches (Google and Shodan), their aggregation, and an improvement of the evaluation in combination with an extensive field study.

4 Approach

For providing a comprehensible alternative to classical penetration testing tools for vulnerability analysis, an approach was developed using both general-purpose and subject-specific search engines in aggregation. First, the necessary raw data, i.e. the banner information, was collected, which will be explained in Sections 4.1 and 4.2 for both search engine alternatives. To extract the raw data from Google, dorks were used. Next, we introduce the “Banner-CPE-CVE”[10] approach including optimizations in Sections 4.3 and 4.4. In Section 4.5, we present the automatic validation approach for evaluating our proposed method and finally, Section 4.6 describes our developed evaluation prototype.

4.1 Dork Generation

A number of dork collections exist on the Internet. In addition to complete websites dedicated to this topic, dorks are also found in blogs and Internet forums. We analyzed these sources and found that they mostly contain redundant data. The work was therefore narrowed down to the two most popular and widely used representatives: the “Google Hacking Database (GHDB)”[11]; now also known as GHDB reborn[12] with about 1,500 entries and the “ExploitDB”[13] with about 4,000 entries. For evaluation, existing errors of the individual dorks in syntax (e.g. missing or incorrectly placed quotation marks) and semantic (e.g. `intext:` instead of `inurl:`) were corrected. However, our test with the use of the

[10] CPE stand for Common Platform Enumeration and CVE stand for Common Vulnerabilities and Exposures.
[11] `https://www.exploit-db.com/google-hacking-database-reborn/`
[12] This indicates that updates are no longer managed by Long [Long et al. 2007].
[13] `https://www.exploit-db.com`

improved dorks showed that still many *false positives*[14] are generated, and the severity of the findings was mostly low. Apparently, existing entries are outdated or inappropriate, and Google may have changed its responses. Unfortunately, the behavior can no longer be checked since Google is available as a web service only in the current version. However, examples of Long et al. [Long et al. 2007] show higher quality results. The insufficient outcomes result not only from the poor quality of the dork databases, but they are potentially also caused by the optimized Google input interface, which tries to defeat dorking attempts.

As the implementation of Google cannot be changed, we improved the dork quality by influencing factors such as the use of extended Google search facilities. The goal of such optimizations is to make the best use of the return volume of the used Google Custom Search interface, which returns of a maximum of 80 entries per request, by bringing high-quality results into this area. This is achieved by improving the precision of the dork specifications. Product names and their version numbers as well as their context can be used for this purpose. The following example shows the application of this approach for an Apache HTTP server. First, the product and the version are specified as precisely as possible. This information is found in the body of a web page. The dork looks as follows:

```
intext="Apache/2.0.63"                                    (3)
```

Second, it was determined with this example that real findings are mostly related to a directory listing. This context information can be used to further refine the dork as follows:

```
intitle="Index of /"                                      (4)
```

Finally, the dork must be restricted further to the domain to be checked. This is independent of the optimization, but is part of the dork. The final version then looks like this:

```
intitle="Index of /" intext="Apache/2.0.63" site:test.de  (5)
```

But the Apache HTTP server alone is currently available in more than 150 versions, all of which are to be queried individually. With the Google Custom Search, only 100 queries can be issued per user and day. Therefore, the method cannot practically be used in this manner in practice, but a compromise of accuracy and applicability is required. This alternative no longer addresses the exact version, but only the root version[15]. For Apache HTTP server, only four root

[14] In our case *false positives* are vulnerabilities determined by the approach, which are not present or no vulnerability in reality.
[15] Apache HTTP Server version 2.0.63 has the root version 2.0.

versions remain (1.3, 2.0, 2.2, and 2.4). Thus, the test could also be carried out practically, and it reduced the *false positive* rate from 90 to about 65 percent.[16]

4.2 Collection of Raw Data using Shodan

To determine raw data for the vulnerability detection, Shodan provides two different request methods via a representational state transfer (REST) interface. First, there is the so-called "host-search" method; Queries can be used similar to searching dorks with general-purpose search engines. Requests for a domain (`hostname:`) or for individual IP addresses or networks (`net:`) are possible. Additionally the so-called "host" method for searching information along individual IP addresses is offered.

Both methods return their results in JSON format. At first glance they look the same. However, when comparing the results in detail, the results of the host method have two additional attributes, which include vulnerabilities and the complete start webpage, respectively.

Therefore, an approach was chosen for the demonstrator that uses the host-search method for domain queries to retrieve available information. Subsequently, the IP addresses of the ascertained hosts are extracted from the results and a new search is carried out using the host method to gain additional attributes. The approach also offers the possibility to search for domains or directly for IP addresses. This is not currently required but could be helpful in the future.

4.3 Banner-CPE-CVE Approach

The basis for "Banner-CPE-CVE" approach is the aggregated usage of the introduced Google and Shodan raw data. In particular, so-called banner information is required for the determination of vulnerabilities (CVEs). For the raw data extraction, Shodan search queries and Google dorks are used as described in the previous sections.

The determination of vulnerabilities from retrieved banner information is a common practice used by several products as well as in several research prototypes. It requires a unique identification of the system or software under consideration (CPE). As operating systems and software developers not always use a common denotation for their products, it is not a trivial task to identify the current system. For improving the CPE-detection quality, the banner recognition integrated in Nmap was examined by Schmidt [Schmidt 2015]. This study revealed coverage gaps due to the fast evolving and widespread landscape of software. In summary, the detection quality could not be improved using this approach compared to the CPE extraction done by Shodan itself. The total number of identified CPEs was similar. But as the relative complements

[16] The number of results was not large so that we could check the results manually.

($CPE_{Shodan} \setminus CPE_{Nmap}$ and $CPE_{Nmap} \setminus CPE_{Shodan}$) also contained results, Shodan seems to use its own method for detecting CPEs. Genge et al. [Genge and Enăchescu 2015] describe their own method for determining CPEs based on banner information. However, their intention primarily focuses on quantity and performance. Furthermore, their ShoVAT tool is not publicly available for further testing. This led us to the conclusion to develop our own solution approach for extracting CPE information based on retrieved banner information, shown in Figure 2. The example demonstrates the determination of a vulnerability for an OpenSSH service in version 4.3.

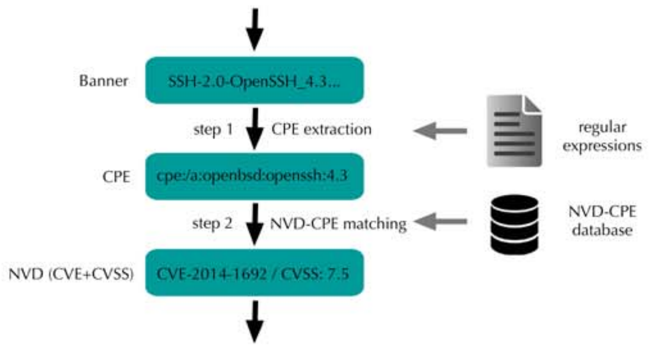

Figure 2: Banner-CPE-CVE approach

The relation between banner information and vulnerability could be determined using regular expressions for identifying the software as well as its version (Step 1). Using this technique, considerably better results were achieved compared to the extraction of CPEs done by Shodan. The concept of the mapping database used is illustrated in the following example, showing first a banner of the OpenSSH server.

```
SSH-2.0-OpenSSH_6.6.1p1 ...                    (6)
```

This banner is hard-coded in the source code of OpenSSH, and therefore not changeable without recompilation. The appropriate regular expression for detecting the CPE is shown in Example 7.

```
SSH-[0-9.]+-OpenSSH_([5-9][0-9.]+)([a-z0-9]*)          (7)
```

In this example, grouping requires two groups: the first group $G_1 =$ `6.6.1` identifying the version and the second group $G_2 =$ `p1` giving the concrete patch level of the OpenSSH service. In combination, the following level 5 CPE is obtained:

$$\texttt{cpe:/a:openbsd:openssh:6.6.1:p1} \tag{8}$$

In case of an empty second group G_2, a level 4 CPE is automatically generated, due to the missing patch level information. Nevertheless, the association of detail information with the appropriate CPE levels is not always unambiguously: Before version 5 of the OpenSSH service, the second group G_2 is not assigned to the level 5 CPE, but in contrast only to level 4. Therefore, two separate entries are necessary for uniquely identifying OpenSSH. These example entries are shown in Table 1.

Table 1: Determination of multi-level CPE: OpenSSH example

CPE	regular expression
`cpe:/a:openbsd:openssh:`	`SSH-[0-9.]+-OpenSSH_([1-4][a-z0-9.]+)`
`cpe:/a:openbsd:openssh:`	`SSH-[0-9.]+-OpenSSH_([5-9][0-9.]+)([a-z0-9]*)`

The mapping of the detected CPE entries to vulnerabilities as shown in Step 2 of Figure 2 requires the NVD Data Feeds[17] provided by NIST. Besides the vulnerability itself, those data feeds contain additional information, such as the mapping of CPE and CVE entries and a description of the vulnerability severity, based on Common Vulnerability Severity Score (CVSS). An example mapping for the Apache httpd service is shown in Table 2.

Table 2: Mapping between CPE and CVE-/CVSS entries (Apache HTTPd)

CPE	CVE	CVSS
`cpe:/a:apache:http_server:1.3.6`	CVE-2000-1205	4.3
	CVE-2001-1449	7.5
	...	...
	CVE-2013-2249	7.5

For the CPE-CVE mapping, logical connections, *AND* or *OR*, are used to combine several CPE entries for one CVE. The sole usage of *OR* connections for level 3 CPEs is ignored intentionally for reducing the *false positive* rate of determined vulnerabilities.

4.4 Context Extension

The results of Shodan queries are structured in so-called modules. Each of these modules includes one service, such as HTTP, regardless under which port this

[17] Structured collection of vulnerabilities; `https://nvd.nist.gov/download.cfm`

service was detected, e.g. HTTP services are commonly offered on port 80/TCP, 8080/TCP or self-defined. As presented, the CPE detection depends on the banner returned for this service probes. If the banner does not reveal the product under consideration, it might lead to wrong or missing CPEs. Especially for the detection of the commonly used DNS server "ISC BIND", only the concrete product version, e.g. "`9.8.1-P1`" is returned as banner and stored in the appropriate module by Shodan. In addition, a module assignment takes place, which can be "dns-udp" or "dns-tcp" depending on the transport protocol used. By using a global mapping of banners to CPE entries, this version number will not only apply to the ISC BIND service, but also to others with similar information.

Table 3: Context extension for CPE determination using regular expressions

Context	CPE	regular expression
`.*`	`cpe:/a:apache:http_server:`	`Apache[/]{0,1}((?:[0-9]+.[0-9.]+)*)`
`dns-.*`	`cpe:/a:isc:bind:`	`([0-9.]+)[/-]*((?:[a-zA-Z]+[0-9])*)`

By introducing a context-specific restriction when processing individual entries of the mapping file (banner to CPE identification), ISC BIND Servers can also be detected without collision. This is done with the introduction of a new attribute per entry, which optionally allows module restrictions of Shodan. Table 3 shows the use of the context extension, which is applied in the second row. In the example, this is "`dns-.*`". To capture both the dns-udp module and the dns-tcp module.

4.5 Automatic validation

Our objective is an automatic validation of the results with the use of the complete test domain for the determination of the *Precision* (for details see Section 5.1). For this end, the contactless test method must be combined with a contact-based validation. There are different approaches for this, some of which must be directly excluded for legal reasons: Running a complete penetration test requires the explicit allowance of the domain owner. Therefore it was not possible to perform a Nessus test of the entire test domain. However, a dosed Nmap deployment (restriction on conspicuous IP addresses and ports) without probing for concrete vulnerabilities, resulting in a behavior close to normal communication, seems viable and provides the basic information needed for an automatic validation. However, Nmap does not provide all required banner information[18],

[18] We used the Nmap Scripting Engine (NSE) "banner" script and got only very limited results, especially in the area of SSL/TLS services, possibly depending on the operating system.

and according to Schmidt [Schmidt 2015], it has only limited CPE detection capabilities.

Thus, we carried out a similar dosed test using the standard library of Python. Using this approach, vulnerable IP/port combinations were directly contacted and the corresponding banner information was read out. These banners were compared with the data determined by Google or Shodan. However, the retrieval of banner information is protocol-specific. For instance, the data on SSH must be retrieved in other ways that information about HTTP.

For example, HTTPS errors occur if the server is addressed with an IP address that is not part of the certificate. This check function must be bypassed by deactivation so that the corresponding banner information can be read out. In addition, it is also necessary to use outdated and unsecure encryption methods[19] – so-called cipher suites – in order to be able to contact outdated systems. The HTTPS problems have been mostly solved. However, minor incompatibilities remain caused by the SSL/TLS implementation of Python and the underlying operating system and their interplay. In addition, no data can be checked that are not based on banner information. This is the case, for example, with DNS servers. Here Shodan retrieves the version and product information with a special query method and stores it as if it were a common banner information.

The measurement error of our the developed validation method was determined to estimate the impact of the problem. This was carried out together with the quality assessment and the same procedure and test domains were used (for details see Section 5). For each test domain, the number of port-based connectivity attempts caused by weaknesses in the test method ($Port_{error}$) was set in relation to the total checks ($Port_{total}$). The total error rate is calculated as follows:

$$F_{total} = \frac{1}{n}\sum_{i=1}^{n}\frac{Port_{i,error}}{Port_{i,total}} = \frac{1}{n}\sum_{i=1}^{n}F_i = \frac{1}{10}\sum_{i=1}^{10}F_i = 0.008 \qquad (9)$$

Thus, in this specific test our validation method has a total error rate of 0.8 percent. This means that of 1000 port-based tests on average only 8 checks fail. For validation of our approach, this seems appropriate. If standard protocols as well as securely configured servers are used, the error rate is even smaller.

4.6 Evaluation Prototype

Our evaluation prototype Contactless Vulnerability Exploration (ClVE), which implements the outlined approach, is presented below. The implementation was done in Python. In the architecture of the implementation, care was taken to

[19] For example, cipher suites that include RC4-based methods.

ensure that different software modules were separated as cleanly as possible from each other, so that they could also be executed in isolation.

Figure 3 shows the schematic structure of the prototype, which is divided into four steps: Collection, Extraction, Identification and Validation. Each step is assigned one or more implementation modules – so-called stages – which can be executed individually. The abbreviations G for Google and S for Shodan are used for the stage names. Stages marked in this way cannot be used generically. In the validation part, the abbreviation B is used for built-in und N for Nessus. Data flows (web or file-based) are displayed vertically with gray and control flows horizontally with black arrows, respectively.

The implementation requires as input the domains to be checked. At the end of the identification and checking process steps, results are output in JSON format. For statistical purposes, there are other output artifacts, such as the representation of raw data. These are not shown in Figure 3.

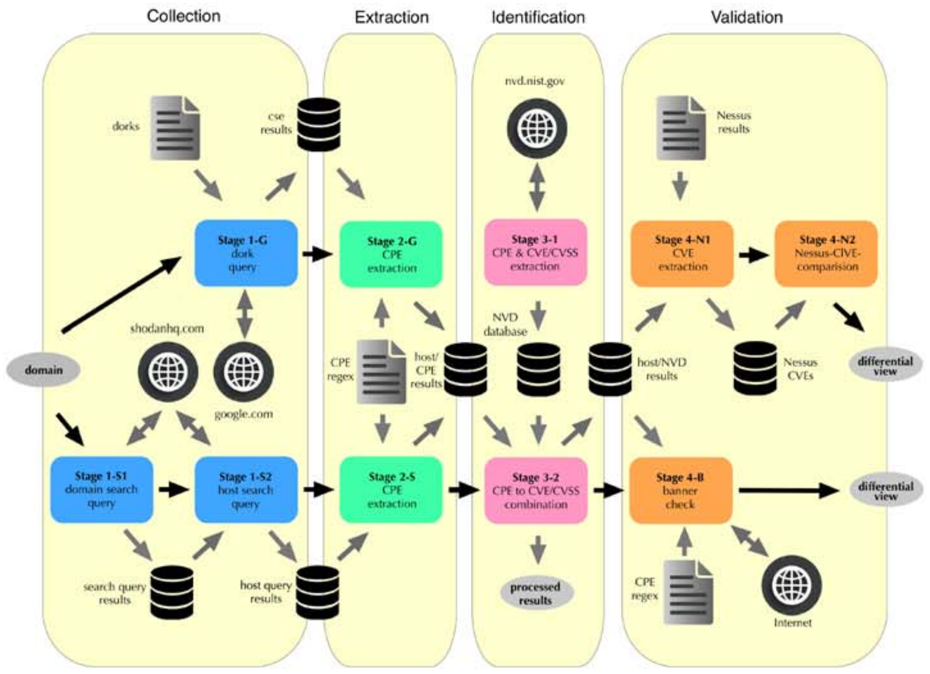

Figure 3: Prototype: Contactless Vulnerability Exploration (ClVE)

The individual stages shown in Figure 3 are presented together with their corresponding steps:

- Stage 1-S1 (Collection) calls Shodan, using a host-search request, to provide all the information that Shodan has stored for the requested domain. The result is returned in JSON format and is structured by IP addresses and ports. Each entry contains a series of information items, including the attributes CPE entry, banner, and domain.
- Stage 1-S2 (Collection) extracts the IP addresses from the result files of Stage 1-S1. Using the host method, the information is retrieved again from Shodan to obtain extended attributes, such as vulnerabilities (CVEs). The result is stored in a file in JSON format for each IP address.
- Stage 1-G (Collection) retrieves vulnerability information from Google using custom dorks. The Google Custom Search Engine is used for this purpose. Retrieved information is transferred to the same JSON structure that is also used for Stage 1-S2.
- Stages 2-S and 2-G (Extraction) determine CPE entries from the obtained raw data. In addition, a self-developed and regular expression based detection is used. In case of Stage 2-S, this provides more and higher quality results than Shodan itself.
- Stage 3-1 (Identification) is a support module. Here the NVD data feeds of the last n years[20] will be downloaded from NIST and converted to a suitable format. Thus, the mapping between CPE identifier and CVE/CVSS entries can be established later.
- Stage 3-2 (Identification) finally combines the obtained CPE information from Stages 2-S and 2-G with the vulnerabilities (CVE/CVSS) from Stage 3-1. The results are stored in a file in JSON format, thereby structuring the vulnerabilities on IP address and port basis.
- Stage 4-B (Validation) contacts the candidates with potential vulnerabilities and retrieves their banners. From this, CPE identifiers are determined using the same mapping file that is also used in Stage 2-S and 2-G. These identifiers are then compared to the original results, and it is determined whether they are *false positive* or *true positive* compared to the banner.
- Stage 4-N1 (Validation) extracts the hosts and vulnerabilities determined by reference tests with Nessus[21] from the results in HTML format and stores them, grouped according to host, port, and module. The structure is similar to the file structure of Stage 3-2.

[20] This setting is configurable; Unless otherwise specified, the default setting is used. This will take into account all available NVD data feeds.

[21] Please note that while recall cannot be checked with the approach from Stage 4-B (cf. Section 4.5), Nessus is used to determine the *relevant elements*.

– Stage 4-N2 (Validation) determines the relevant information (host, port and vulnerability) from the raw data of Stage 3-2 and Stage 4-N1 during preprocessing. The two sets of data are compared so that the *false negative*[22] rate can be determined.

Despite the prototypical implementation of ClVE, the tool shows sufficient robustness so that the tests could be carried out sucessfully.

5 Evaluation

In accordance with the literature (e.g., Tung et al. [Tung et al. 2013] or Mohammed [Rawaa 2016]), we will use *Precision* and *Recall* as criteria for quality evaluation. In addition, other parameters are also important, e.g. the age of the Shodan and Google raw data, and the runtime of the approach in comparison to Nessus.

To obtain an unbiased selection of test candidates and to achieve a good sample coverage, we chose candidate domains from the ten categories of gross value creation[23], which are also used to determine gross domestic product (GDP), including "construction, agriculture, and forestry", "industry", and "construction" (cf. Table 4). We have explicitly chosen an extensive field trial in order to cope with real problems and to cover a large test area. For self-built tests settings, it is practically impossible to set up several thousand systems with different IP addresses and configurations. In addition, some search engines already recognize well-known test software, for instance Shodan honeypots.

One domain was randomly selected per category. Chronologically, the complete selection of all domains took place before the first validation run. Once selected, domains were no longer changed to ensure independence of choice. The domains thus determined originate primarily from Germany. Since these domains serve only as random test objects and since potential attacks based on findings of vulnerability are not to be promoted, the domains are not listed. Rather, the test candidates are purely for the quality assessment of the test method.

5.1 Precision and Recall

The *Precision*, defined as ratio of *true positives* and *all positives*, was automatically determined using the evaluation method described in Section 4.5. Within the scope of the evaluation, only systems with potential vulnerabilities were contacted. The banner is retrieved for each system and the resulting CPE entries are

[22] We defined *false negatives* as vulnerabilities that are present, but can not be determined by our approach.

[23] `http://ec.europa.eu/eurostat/statistics-explained/index.php/National_accounts_and_GDP`

determined. This set is called CPE_{Stage4}. For comparison, the CPE identifiers (denoted by CPE_{Stage3}) from which the CVE identifiers were derived in the extraction phase are used. In order not to tamper the result, newly determined CPE entries must be eliminated with the operation $CPE_{Stage3} \cap CPE_{Stage4}$.

To determine the *Precision*, ten different domains were used from the categories listed in Table 4. The weighting per domain with the factor of 0.1 (this corresponds to an equilibrium) was defined before the quality validation was carried out. This also applies to the quality evaluations carried out during the further course of the work. This results in the following calculation, where i denotes the different domains:

$$\begin{aligned} Precision_i &= \frac{true\ positives_i}{true\ positives_i + false\ positives_i} \\ Precision_i &= \frac{|CPE_{i,Stage3} \cap CPE_{i,Stage4}|}{|CPE_{i,Stage3}|} \\ Precision &= \frac{1}{n}\sum_{i=1}^{n} Precision_i = 0.821 \end{aligned} \tag{10}$$

This value of *Precision* means that 82.1 percent of the findings are real vulnerabilities (see Table 4 for details). However, in the case of the CPE determination, it must be noted that the determined product and version information does not have to correspond to the actually installed software packages. For example, they may use so-called backports. This potentially results in reporting more vulnerable systems compared to reality. However, conventional test methods are also subject to this measuring error, so that the comparison is fair.

Table 4: Results for *Precision* are listed by category

Categories	$\boldsymbol{Precsision_i}$ (in percent)
Agriculture, forestry and fishing	100.0
Industry	50.0
Construction	100.0
Services (Trades, transport, food, ...)	40.0
Information and communication	100.0
Financial and insurance activities	66.7
Real estate activities	100.0
Services (scientific, technical, support, ...)	85.4
Public administration, defense, education, ...	86.8
Other services	92.1

As already described, the determination of *Recall*, defined as the ratio of the *true positives* to *relevant elements* is difficult, because the actual number of vulnerabilities of the object to be evaluated is unknown and cannot be determined reliably without manual introspection of the systems. Therefore, an evaluation with respect to real and unknown systems was unfeasible in the field study and instead, the *Recall* was determined in comparison to classical penetration testing tools such as Tenable Nessus. For this reason, several assumptions had to be made:

- For achieving at least an upper bound with respect to classical penetration testing systems, it was assumed, that findings determined by Nessus provide the set of *relevant elements.*
- The domain for comparison was restricted to a subdomain with 768 IP addresses. For this subdomain, an automated evaluation run with Nessus including its evaluation was possible[24].
- The interval between test and reference measurements was almost six months. According to the operators of the subdomain, however, there were only marginal changes. In order not to influence the result by newly discovered vulnerabilities, only NVD data feeds were used which were available at the time of the Nessus test.
- In an automated comparison of our approach against Nessus, only the CPE-based findings were used, since no qualitative comparison was possible for the remaining Nessus findings. Those findings include for instance configuration vulnerabilities and information disclosure, and supply 35.9 percent of the total findings. A comparison with CVE entries is not useful since Nessus only outputs selected entries.

The reconciliation between the two data sets was carried out automatically (Stage 4-N1 and Stage 4-N2). During the Nessus test run, 91 CPE entries (*relevant elements*) could be determined. ClVE returned 45 results (*true positives*), which could also be found with Nessus ($CPE_{Nessus} \cap CPE_{ClVE}$). This results in:

$$Recall = \frac{true\ positives}{relevant\ elements} = \frac{|CPE_{Nessus} \cap CPE_{ClVE}|}{|CPE_{Nessus}|} = \frac{45}{91} = 0.495 \quad (11)$$

The *Recall* means that 49.5 percent of the actual findings were determined using the method according to the estimation. Due to the restricted test quantity and the assumptions made, the value is only of limited use. We analyzed the results that only Nessus could determine ($CPE_{Nessus} \backslash CPE_{ClVE}$) and found that these are primarily from the operating system. Nessus determines these

[24] The operator has granted an inspection permit.

mainly from the ping behavior and not from banner information, so that they are outside the recognition area of ClVE. If these results are excluded in the calculation of the recall, a value of about 90 percent is achieved.

5.2 Age of the Raw Data

Neither the respective websites nor the literature provides information about the age of the raw data of Google and Shodan. Therefore, timestamps of Shodan-based data were evaluated to check the timeliness of the determined data. Google-based age information is not available.

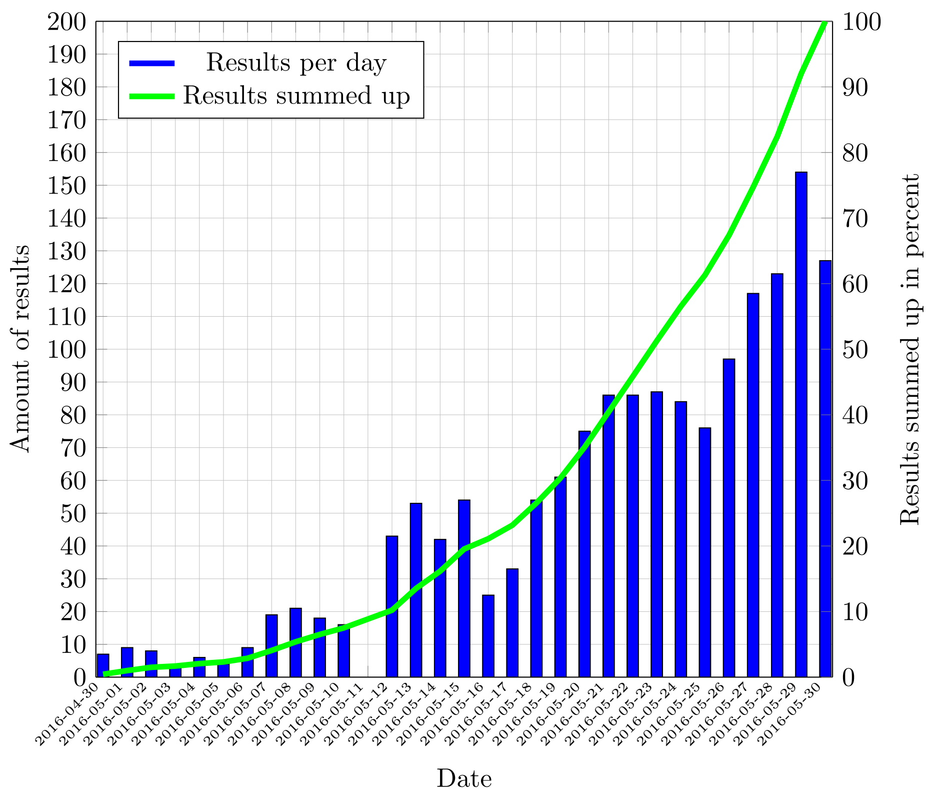

Figure 4: Age of the Shodan raw data

Figure 4 shows the time stamp information for the category “public administration, defense, education, ...” in detail. Half of the data is one week old and 75 percent of the data is at most two weeks old.

To determine the average maximum age of the raw data, the domains from the ten presented categories were used. The average of the maximum age is calculated as follows:

$$t_{max} = \frac{1}{n}\sum_{i=1}^{n} t_{i,max} = \frac{1}{10}\sum_{i=1}^{10} t_{i,max} = 25 \tag{12}$$

Thus, the average maximum age (t_{max}) is 25 days, with individual values ranging from 15 to 31 days. The average age of 25 days is reasonable for this work, but might present a problem with rapidly changing environments.

5.3 Shodan vs. ClVE Results

The following comparison illustrates the improved detection capability of ClVE compared to Shodan. To this end, the CPE and CVE entries that were already present in the Shodan raw data are compared with those which could be extracted with ClVE. For this purpose the CPE recognition rate is determined. Level 3 CPE entries are ignored as these return almost always *false positives*. Thus, the relevant CPE entries $CPE_{i,Shodan,rel}$, and $CPE_{i,ClVE,rel}$ can be calculated as follows. From this, the CPE recognition rate ER_{CPE} of ClVE with respect to Shodan is determined.

$$CPE_{i,Shodan,rel} = \sum_{Level=4}^{7} CPE_{i,Shodan,Level} \tag{13}$$

$$CPE_{i,ClVE,rel} = \sum_{Level=4}^{7} CPE_{i,ClVE,Level} \tag{14}$$

$$ER_{CPE} = \frac{1}{n}\sum_{i}^{n} \frac{CPE_{i,ClVE,rel}}{CPE_{i,Shodan,rel}} = \frac{1}{n}\sum_{i}^{n} ER_{i,CPE} = 1.35 \tag{15}$$

Thus, on average ClVE identifies 35 percent more CPE entries than Shodan within the tested domains.

When analyzing the interim results from Table 5 and comparing them with Shodan output, it can be seen that ClVE provides at least as many results as Shodan, except in one case – in the domain of service providers. After a closer look at the detected CPE entries, Shodan discovered a total of 286 hardware components from Fortinet. Unfortunately, it could not be finally clarified, how this could be realized, because it was not possible solely from banner information.

The following equation is used to calculate the CVE recognition rate ER_{CVE} related to Shodan:

$$ER_{CVE} = \frac{1}{n}\sum_{i=1}^{n} ER_{i,CVE} = 11.2 \tag{16}$$

Table 5: CPE detection rate results

Categories	**CPE detection rate**		
	CPE_{Shodan}	CPE_{ClVE}	ER_{CPE} in [%]
Agriculture, forestry and fishing	1	1	100.0
Industry	2	2	100.0
Construction	1	2	200.0
Services (Trades, transport, food, ...)	4	10	250.0
Information and communication	1	1	100.0
Financial and insurance activities	3	3	100.0
Real estate activities	1	1	100.0
Services (scientific, technical, support, ...)	781	582	74.5
Public administration, defense, education, ...	646	1072	165.9
Other services	851	1363	160.2

The result shows that ClVE extracts (averaged over the ten test domains) 11.2 times as many CVE entries as Shodan. The result is no surprise, as Shodan currently extracts only few CVE entries.

Table 6: CVE detection rate results

Categories	**CVE detection rate**		
	CVE_{Shodan}	CVE_{ClVE}	$ER_{i,CVE}$
Agriculture, forestry and fishing	2	40	20,0
Industry	22	6	0.3
Construction	4	8	2,0
Services (Trades, transport, food, ...)	15	19	1.3
Information and communication	3	10	3.3
Financial and insurance activities	27	59	2.2
Real estate activities	2	3	1.5
Services (scientific, technical, support, ...)	491	4700	9.6
Public administration, defense, education, ...	530	15353	29,0
Other services	283	12095	42.7

Table 6 shows the individual results. The analysis reveals that the detection rate of ClVE was lower than that of Shodan in only one case (see category “industry”). This is due to the fact that Shodan determines CVE entries also from SSL/TLS data and not only from the banner. In all 22 cases, the additional finding was CVE-2016-0160, the Heartbleed bug[25].

ClVE was able to achieve better results in both CPE and CVE detection than Shodan. However, the analysis of the results reveals that Shodan determines some products and vulnerabilities in an alternative way, which cannot be determined by the method used in ClVE.

[25] `https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-0160`

5.4 ClVE Results, Shodan versus Google

In order to compare the findings of the Shodan- and Google-based approach, the CPE entries identified were quantitatively compared. To this end, the ten different domains were used as test candidates. The share of the Google-based CPE entries E_{Google} from the total result is calculated as follows:

$$\begin{aligned} E_{i,Google} &= \frac{|CPE_{i,Google}|}{|CPE_{i,Google}| + |CPE_{i,Shodan}|} \\ E_{Google} &= \frac{1}{10}\sum_{i=1}^{10} E_{i,Google} = 0.003 \end{aligned} \tag{17}$$

Thus, 0.3 percent of the CPE entries were identified using the Google-based approach. The raw data $E_{i,Google}$ can be obtained from Table 7. Qualitatively, most of the Google results obtained are level 4 CPE entries, additionally about 40 percent level 3 entries, and no level 5 CPE entries. Thus, the share of level 3 CPE indicators is about twice as high as with Shodan (about 20 percent).

To examine the crossover of Google and Shodan results on a CPE entry basis, the intersection $CPE_{i,inter}$ and the resulting percentage of Google share $E_{i,inter}$ is calculated as follows:

$$CPE_{i,inter} = CPE_{i,Shodan} \cap CPE_{i,Google} \tag{18}$$

$$E_{i,inter} = \frac{|CPE_{i,inter}|}{|CPE_{i,Google}|} \tag{19}$$

The comparison of the intersections $E_{i,inter}$ (see Table 7) shows that Shodan determined between 5.9 and 20.7 percent of the CPE identifiers identified with Google. Thus, the Google-based approach provides some results that could not be identified using the Shodan-based approach.

In summary, the Google results are neither qualitatively[26] nor quantitatively convincing. Only three of the ten categories used were able to achieve results with the Google-based approach. Ultimately, the Shodan results were only marginally extended.

5.5 Performance

In the following, the performance of ClVE and Nessus was compared. To this end, a comparison is made between ClVE and Nessus execution runtimes. The aim here is to put the two programs in a rough relationship. For comparison, average run times per host are calculated for ClVE (t_{ClVE}) and Nessus (t_{Nessus}).

[26] level 3 CPE identifiers usually lead to *false positives.*

Table 7: Results for E_{Google}, $|CPE_{inter}|$ and E_{inter}

Categories	**Google share**	**Intersection**	
	E_{Google} [%]	$\|CPE_{inter}\|$	E_{inter} [%]
Agriculture, forestry and fishing	0.0	0	0.0
Industry	0.0	0	0.0
Construction	0.0	0	0.0
Services (Trades, transport, food, ...)	0.0	0	0.0
Information and communication	0.0	0	0.0
Financial and insurance activities	0.0	0	0.0
Real estate activities	0.0	0	0.0
Services (scientific, technical, support, ...)	0.6	1	5.9
Public administration, defense, education, ...	1.2	6	20.7
Other services	1.0	3	17.6

In addition, the reference measurements used in the last sections with ClVE as well as data determined by penetration tests in 2015 with Nessus are used.

$$t_{ClVE} = \frac{1}{n}\sum_{i=1}^{n} t_{host,i} = 11 \text{ s} \tag{20}$$

$$t_{Nessus} = \frac{1}{n}\sum_{i=1}^{n} t_{host,i} = 23 \text{ min} \tag{21}$$

The tests with ClVE can be carried out in a much shorter time than with Nessus. On average, the run time of the test is 11 seconds per host using ClVE, and 23 minutes with Nessus. Thus ClVE is about 125 times faster. For this reason, the alternative approach using ClVE can be particularly used if test results are required fast and in the case of large domains. An example of this would be the determination of a zero-day vulnerability at a university or a blue chip company.

6 Conclusion and Future Work

We have investigated the question of contactless vulnerability analysis of large Internet domains. Building on previous work to use search engine data, we have performed an extensive field analysis and demonstrated that neither general nor subject-specific search engines deliver enough data, i.e., that neither Google-based nor Shodan-based vulnerability analysis yields sufficient coverage. We have proposed a new approach based on refining the search terms, combining results from both search engines, and augmenting the results by pairing them with data from publicly available vulnerability databases. We evaluated our approach in a field study and demonstrated that even in comparison with contact-based analysis, i.e. using Nessus, our approach performs well both in terms of precision and recall, is much faster and less stressful to the network environment. Thus, with our approach, contactless vulnerability analysis has achieved a reasonable level of maturity to enhance and complement classical analysis based on direct, contact-based methods. In addition, it enables analysis of domains with unclear legal audit status, as the target systems are not directly contacted.

Current limitations of the approach are the constrained availability of raw data with Shodan and the lack of configuration error detection. Future work will comprise the extension of contactless vulnerability detection with respect to new sources of information[27]. This also includes vulnerabilities that cannot be mapped to CVE identifiers or that cannot be assigned to any CVE identifier. To this end, a method has to be developed that enables a clear description of these vulnerabilities in order to establish a comparability (for example, with Nessus) and to determine the quality of the search results. A first approach would be to use common configuration enumeration (CCE) to describe potential configuration problems.

References

[Billing et al. 2008] Billig, J., Danilchenko, Y., Frank, C. E.: "Evaluation of Google Hacking"; Proceedings of the 5th Annual Conference on Information Security Curriculum Development (pp. 27-32), InfoSecCD '08, Kennesaw, Georgia, 2008.

[Bodenheim 2014] Bodenheim, R. C.: "Impact of the Shodan Computer Search Engine on Internet-facing Industrial Control System Devices"; Master's Thesis. Air Force Institute of Technology, Graduate School, Ohio, Mar. 2014.

[Dalek et al. 2013] Dalek, J., Haselton, B., Noman, H., Senft, A., Crete-Nishihata, M., Gill, P., Deibert, R. J.: "A Method for Identifying and Confirming the Use of URL Filtering Products for Censorship"; Proceedings of the 2013 Conference on Internet Measurement Conference, IMC '13, Barcelona, Spain, 2013.

[Erven and Collao 2015] Erven, S., Collao, M.: "Medical Devices: Pwnage and Honeypots"; DerbyCon 2015, Louisville, Kentucky, 2015.

[ENISA 2016] European Union Agency For Network And Information Security (ENISA): "ENISA Threat Landscape 2015"; Jan 2016. https://www.enisa.europa.eu/activities/risk-management/evolving-threat-environment/enisa-threat-landscape/etl2015/etl2015.

[Genge and Enăchescu 2015] Genge, B., Enăchescu, C.: "ShoVAT: Shodan-based vulnerability assessment tool for Internet-facing services"; Security and Communication Networks (pp. 2696-2714), volume 9, 2015. http://dx.doi.org/10.1002/sec.1262.

[Goldman 2013] Goldman, D.: "Shodan: The scariest search engine on the Internet"; CNN Money, Apr 2013. http://money.cnn.com/2013/04/08/technology/security/shodan/.

[McGuffee and Hanebutte 2013] McGuffee, J. W., Hanebutte, N., "Google hacking as a general education tool", Journal of Computing Sciences in Colleges, Volume 28 Issue 4, Apr 2013.

[Imperva 2011] Imperva: "Hacker Intelligence Initiative, Monthly Trend Report #3, August 2011, Hacker Intelligence Summary Report - The Convergence of Google and Bots"; Report, Aug 2011. http://www.imperva.com/docs/HII_The_Convergence_

[27] Censys from the University of Michigan might be a worthwhile candidate, see https://censys.io.

of_Google_and_Bots_-_Searching_for_Security_Vulnerabilities_using_Automated_Botnets.pdf.

[John et al. 2010] John, P., Yu, F., Xie, Y., Abadi, M., Krishnamurthy, A.: "Searching the Searchers with Searchaudit"; Proceedings of the 19th USENIX Conference on Security, USENIX Security'10, Washington, DC, 2010. http://dl.acm.org/citation.cfm?id=1929820.1929832.

[Kohl 2016] Kohl, M.: "Verwundbarkeitsanalyse mittels Google Dorks unter der Verwendung von Common Platform Enumeration (Vulnerability analysis using Google Dorks and common platform enumeration)"; Bachelor's Thesis, TU Kaiserslautern, Feb 2016.

[de Kunder 2016] de Kunder, M.: "The size of the World Wide Web (The Internet)"; http://www.worldwidewebsize.com.

[Eireann 2011] Leverett, É.: "Quantitatively Assessing and Visualising Industrial System Attack Surface"; PhD thesis, University of Cambridge, Jun 2011.

[Long et al. 2007] Long, J., Gardner, B., Brown, J.: "Google Hacking for Penetration Testers, Volume 2"; Burlington, MA, Syngress, Aug. 2007. ISBN: 978-1-59749-176-1.

[Rawaa 2016] Rawaa, M.: "Assessment of Web Scanner Tools"; International Journal of Computer Applications (0975-8887), Volume 133, No.5, Jan 2016.

[European Union 2013] European Union: "Directive 2013/40/EU of the European Parliament and of the Council"; 2013. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2013:218:0008:0014:en:PDF.

[Opp 2014] Opp, A.: "Schwachstellenanalyse mittels klassischer Internet-Suchmaschinen (Vulnerability analysis using classic Internet search engines)"; Master's Thesis, Hochschule Kaiserslautern, Oct. 2014.

[Oswald 2015] Oswald, M.: "Verwendung von Google Dorks zur Durchführung von anonymisierten und personalisierten Massensuchanfragen (Use of Google dorks to perform anonymous and personalized mass queries)"; Master's Thesis, FernUniversität in Hagen, Sep. 2009.

[PWC 2016] PWC: "Turnaround and transformation in cybersecurity"; Jan. 2016. http://www.pwc.com/gsiss.

[Scherer 2008] Scherer, B.: "Anpassung von Penetrationstests auf die speziellen Gegebenheiten der Fraunhofer-Gesellschaft (Adaptation of penetration tests to the specific circumstances of the Fraunhofer-Gesellschaft)"; Diplomarbeit, Fachhochschule Kaiserslautern, Jan. 2008.

[Schmidt 2015] Schmidt, O.: "Verwundbarkeitsanalyse mittels themenfeldorientierten Suchmaschinen (Vulnerability analysis using subject-oriented search engines)"; Master's Thesis, FernUniversität in Hagen, Sep. 2015.

[Sedlmeier 2016] Sedlmeier, M.: "Klassifizierung und Evaluation verschiedener Datensammler zur Schwachstellenermittlung (Classification and evaluation of different data collectors for vulnerability analysis)"; Master's Thesis, FernUniversität in Hagen, Jul. 2016.

[Simon 2016] Simon, K.: "Vulnerability Analysis Using Google and Shodan"; In International Conference on Cryptology and Network Security (pp. 725-730). Springer International Publishing. Nov. 2016.

[von Thaden 2015] von Thaden, S.: "Analyse und Optimierung von Dork-Anfragen (Analysis and optimization of dork queries)"; Master's Thesis, FernUniversität in Hagen, Sep. 2015.

[Toffalin et al. 2016] Toffalini, F., et al., "Google dorks: Analysis, creation, and new defenses" Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 255-275). Springer International Publishing, 2016.

[Tung et al. 2013] Tung, Y.-H., Tseng, S.-S., Shih, J.-F., Shan, H.-L.: "A cost-effective approach to evaluating security vulnerability scanner"; Network Operations and Management Symposium (APNOMS), 15th Asia-Pacific, Hiroshima, Japan, Sep. 2013.

[U.S. DHS 2012] U.S. Department of Homeland Security: "ICS-CERT Monitor October/November/December 2012"; 2012. `https://ics-cert.us-cert.gov/sites/default/files/Monitors/ICS-CERT_Monitor_Oct-Dec2012.pdf`.

[Zhang et al. 2015] Zhang, J., Notani, J., Gu, G.: "Characterizing Google Hacking: A First Large-Scale Quantitative Study"; International Conference on Security and Privacy in Communication Networks. 10th International ICST Conference, SecureComm 2014, Beijing, China, 2015. ISBN: 978-3-319-23829-6.