The current state of DNS Lame delegations

An analysis of the current state of lame delegations within the Swedish .se tld

Master Security and Network Engineering
University of Amsterdam

Research Project 2
Project Report
Version: 2.1

Author:
Alexander Blaauwgeers
alexander.blaauwgeers@os3.nl

Supervisor:
Arris Huijgen
ahuijgen@deloitte.nl

November 17, 2020

Abstract

This study develops a framework for recognizing lame delegations within the Swedish tld. In order to answer the research question, the author tested the framework in a proof of concept. Based on the results of the research, the study concludes that a limited number of Swedish domains have lame records to unregistered domains. Specifically, it identifies 862 domains with Lame NS delegation and 356 domains with Lame MX delegations. The domains involved may be vulnerable to takeover.

Acknowledgements

I would like to thank my supervisor, Arris Huijgen, for his support during this research project.

Contents

1 Introduction

The Domain Name System (DNS), described by Paul Mockapetris in rfc1034[1], provides a mechanism for naming resources in a way such the human-readable host names are usable on different hosts and networks such as machine-interpretable addresses.

Engineers designed DNS with a focus on scalability rather than security. The security requirement arose with the introduction of the internet to the general public in the mid 1990'-s[2], by which time DNS had already acquired an indispensable position within the internet information structure.

This DNS system is used by email systems to locate the Mail Transfer Agent (MTA) that belongs to the hostname used in the email address, a task carried out by the mail exchanger resource record (MX) within the DNS zone. This MX record specifies which host has the MTA for the domain, and this MTA should accept mail for forwarding to the domain. In RFC 1912, David Barr[3] described errors that are often found within the operation of DNS. One of these errors is lame delegations, defined as delegations of functionality to another entity without that entity in place to serve this request. Lame delegations may impact the availability of services and pose security risks. This paper investigates how many lame delegations exists in the Swedish TLD zone by designing a framework.

1.1 Goal

The goal of this research is to develop a framework for detecting lame delegations within the Swedish TLD. This framework can be used as a starting point for further research.

1.2 Research question

The main research question is:

Are the domains below the Swedish .se tld vulnerable to lame delegation take-over?

This research question can be divided into multiple sub-questions:

1. What are lame delegations and their security implications?
2. How many lame delegations exist within the .se tld?
3. Can we identify any top-talkers[1] among them?

1.3 Scope

This research focuses on lame delegations on the Swedish .se tld.[2]. The focus is on the Swedish .se tld because the .se zone file has been published. Delegations that point from the Swedish .se tld to outside are also within scope. The focus of the project will be delegations which are vulnerable to takeover by others to retrieve email. Third-level records such as "student.mau.se.", that are below the APEX[3] are out of scope, including third level delegations which are below the APEX like CNAME and SRV, and third level zone delegations with NS records.

[1]The term "Top-talkers" refers to a large set of lame delegations sharing to the same delegation endpoint.
[2]ICANN assigned the ccTLD ".se" to Sweden based on ISO_3166-1.
[3]Apex[4]: "The point in the tree at an owner of an SOA and corresponding authoritative NS RRset."

2 Definition and related work

This section describes the background of the research and the definitions it uses to answer the first research question. It also describes the state of the art based on related work.

2.1 Background information

The Domain Name System (DNS) has several essential roles within the internet ecosystem. In addition to its best known function, converting hostnames into IP addresses, it is used as a hierarchical decentralized database[5]. The DNS system is characterized by its delegating nature. In almost all cases, multiple DNS servers are involved in the response to the query of the DNS client.

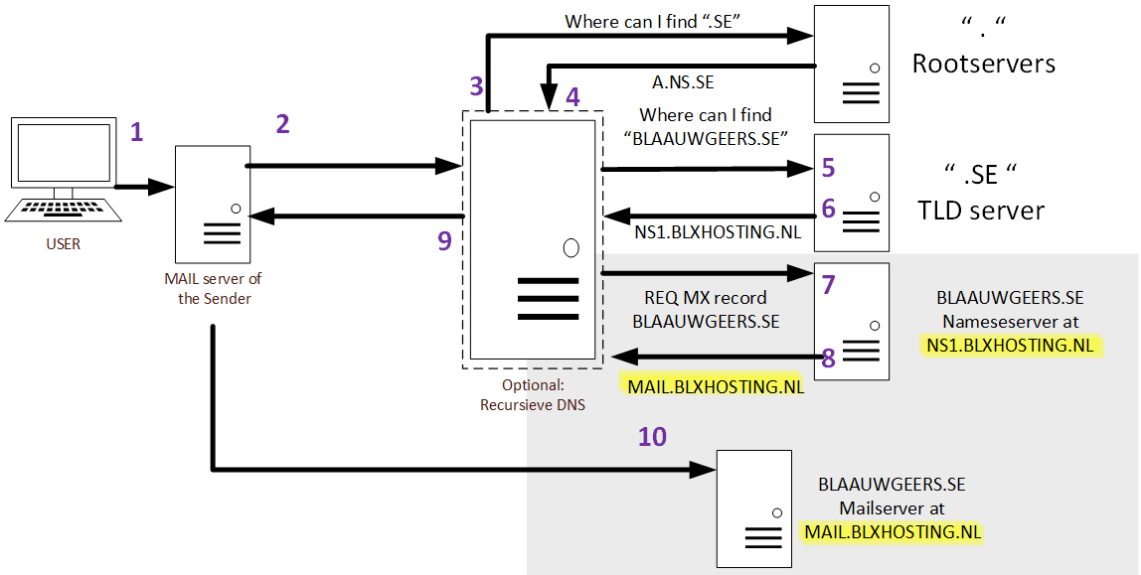

Figure 1: *Example of how DNS is used for email delivery*

Figure 1 depicts the hierarchical and decentralized process by which DNS works for email. The process starts when the user(1) sends an email via SMTP to the mail server.

The header of the email should contain the name of the recipient's mailbox (e.g., username, alias ...) followed by the recipient's domain name, with the @-symbol serving as a separator. The mail server then requests(2) the Mail eXchange (MX) record of the domain name through a recursive DNS server, which can be either external to the mail server or integrated into it.

Next, the recursive DNS server initiates the decentralized process by requesting(3) the rootserver. The answer(4) will be a delegation, via a NameServer(NS) record, which is the delegation of authority to a lower DNS server. In the example, this is the Swedish .se tld operator hosted on, for instance, A.NS.SE on behalf of the Swedish Government by the Swedish Internet Foundation.

The request is then redirected(5) to the DNS server of the operator of the Swedish .se tld. This response(6) is redirected with an NS record delegation to a delegated nameserver. In the example, this answer(6) contains a delegation to "ns1.blxhosting.nl.". The process continues until the DNS server with authority over the domain. This answers the request(7) in the form of an MX record(8).

2.2 Lame delegation

The principle of DNS assumes that all delegations are correct. This research focuses on situations where this is not the case. In many cases, an incorrect delegation includes a broken relationship between parties in the DNS landscape. Within the DNS community, this phenomenon which is described by David Barr[3], this is also called a lame delegation.

In the example of the .se domain "blaauwgeers.se" Figure 1 shows a delegation to blxhosting.nl. It is assumed that the domain blxhosting.nl is not registered, so there will be no response to the request numbered 7 and 10 in the figure. Therefore delegations numbered by 6 and 8 are lame delegations, and thus "blxhosting.nl" is the lame delegation endpoint. The user, in standard configuration, will receive an error message that their email can not be delivered.

2.3 Security implications caused by lame records

In a normal situation, a lame record results in the unavailability of service. The end-user, or the entity who sends the email, receives an error message and the process stops. However, it is possible that a third party intentionally or unintentionally registers the domain to which a lame record refers, in which case the lame delegation will be restored and the new registrant will obtain authority over the delegation and thus control over the domain delegated to it.

In the example scenario shown in Figure 1, this means that someone who registers the unregistered domain 'blxhosting.nl' receives control, or partial control, over the domain blaauwgeers.se. This control will remain in place for as long as the delegation continues to exist. In the event a third party gains control over the delegation, this party can set up a rogue mailserver and thus receive (confidential) emails to the domain. This mail can then be relayed by the rogue mailserver at a later stage, if the third party is able to discover the correct delegation, as such by fixing a typo. Under such a scenario, the data leak might remain unnoticed.

2.4 Partially lame delegation

Within the internet ecosystem, it is common to designate backup servers for when the primary server cannot meet demand. This solution is also advised for DNS-related services. In some cases, only one of the delegated records (e.g. the backup server) is lame, making the delegation only partially lame. As a result, it is possible that the lame delegation remains unnoticed or hidden from the owner of the domain for a longer period of time.

Figure 2 shows an example of a backup lame delegation.In the zone file, one can see three MX records. In this example scenario, it is assumed that the first record to "newhosting.nl" is a proper delegation to a new provider. This delegation is primary because the dns operator has the lowest priority to this record. The second record, we consider as lame, is blxhosting.nl. This might be the former provider. If the first delegation becomes unavailable, the system switches to the next delegation. This can happen when the primary server fails or is overloaded.

This failover scenario, that can occur under usual conditions such as peak load, but it can also be triggered intentionally by a Denial of Service (DoS) attack or a distributed (DDoS) variant of it. A scenario which is not an uncommon attack scenario in the current time. Finally, there is the scenario that the primary delegation is lame and its status is hidden by a well functioning secondary delegation.

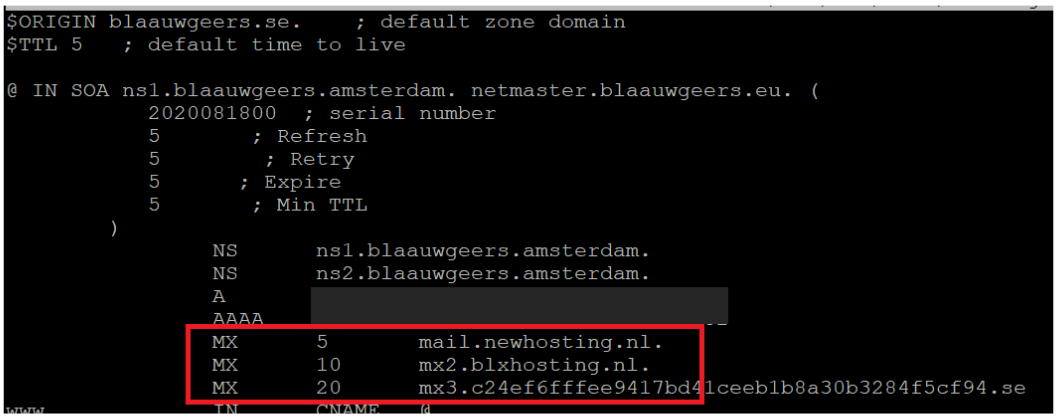

Figure 2: *Example zone file with a partially lame delegation*

2.5 Related work

Amreesh Phokeer [6] has performed a case-study of lame delegations within the reverse DNS Records in the African region. He found that within the African IP space 45% of all reverse domains are lame. However, this research was performed on PTR pointer records instead of delegations used in functions like the mail system.

Pappas[7] has performed a study on the impact of configuration errors in DNS. He classified lame delegations into three cateogires, dending on the type of error found:

1. Type 1: Non responding server
2. Type 2: DNS error indication
3. **Type 3: Non-authoritative answer (!)**

Types 1 and 2 are less relevant this study because they are difficult to take ove and are lame due to server problems other than an unregistered delegation endpoint. These are based on failures or limited rights.

Type 3 lame delegations are when another server, often the dns server of the operator of the top level domain (TLD), responds on behalf of the delegation endpoint, which is a signal that the domain is not available. Some possible reasons for its unavailability include the domain not being registered, the existence of restrictions on its use and quarantine. This theory serves as a basis for the method used in this research.

3 Methodology

In order to answer the research questions, this thesis offers a framework. This framework forms the methodology behind the study and determines whether a DNS record is a lame delegation on a large scale, such as the Swedish .se tld zone. Therefore, the availability of the zone file is a conditional requirement. The framework is based on the operation of the DNS protocol, and the availability of the Swedish zone file. It is made up of stages to create modularity and ease of use. These stages are used in coding the scripts for the proof of concept, due to the large amount of data handled. Each stage can be performed independently, but they are dependent on the input of the previous stage. The modularity of the framework stages makes it possible to repeat stages or add modified stages by, for instance, searching for other records, using a different source like a fixed list of domains.

3.1 Stage 0: DNS zone transfer

The initiation stage involves the gathering of the initial dataset. An important condition of this stage is that the dataset be as complete as possible. The completeness is important because we also want to include less used domain names in the research. The operator of the Swedish .se tld has made the zone file publicly available. This study assumes that the published zone file is correct, though also tries to verify this point by comparing the zone file before and after the author's registration of a domain named "blaauwgeers.se".

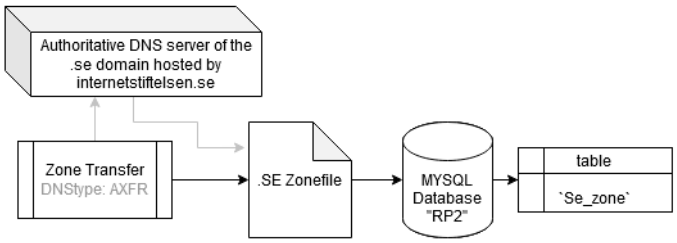

Figure 3: *Zonetransfer*

Figure 3 shows the elaboration of this stage, including zone transfer with DNS type AXFR. The output of this transfer is stored in the database.

IN: `dig @zonedata.iis.se se AXFR`

OUT: `INSERT INTO ‘se_zone‘ (‘id‘, ‘label‘, ‘ttl‘, ‘class‘, ‘rtype‘, ‘data‘)`
`foreach line { VALUES (NULL, 'LABEL.SE', '3600', 'IN', 'NS', 'NS.SERVER.DE') }`

3.2 Stage 1: The authoritative name server of the domains below the Swedish .se tld

In the framework's first stage the author requested the Start of Authority (SOA) records[4] via a direct request from all nameservers as they were registered in the zone file. The result of this request, together with the status code[5] of the answer, will be stored in the database. In addition to retrieving the SOA record, the author requested the MX record of the zone apex and stored it in the database. This process is outlined Figure 4 and in the following pseudocode.

[4] "Exactly one SOA RR should be present at the top of the zone" - [3]rfc1912

[5] The common name for this status code is RCODE as specified in rfc1035 on DNS.

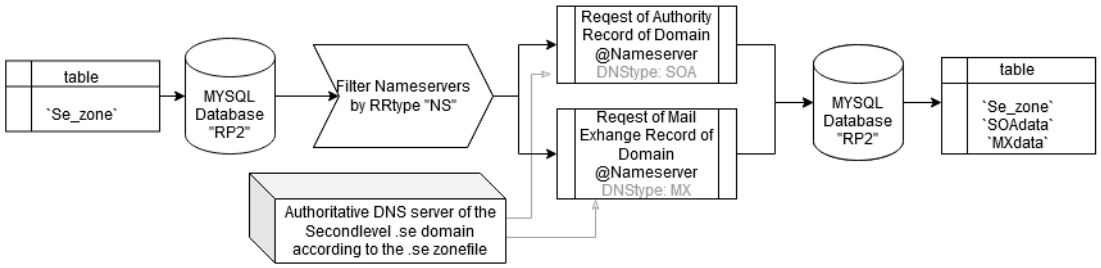

Figure 4: *Request at the authoritative name server of domains below the Swedish .se tld.*

IN: `SELECT ‘label‘, ‘data‘ AS ‘nameserver‘ FROM ‘se_zone‘ WHERE ‘rtype‘ = "NS";`
DO:
foreach record {
`dig $label @$nameserver SOA`
`dig $label @$nameserver MX`
}
OUT: `INSERT INTO ... SOAdata | MXdata`

3.3 Stage 2: Checking the authority of the delegation

The second step, shown in Figure 5, involves checking the start or authority of the delegation endpoint. This study does so for all delegations, though focuses on NS and MX. Records with a status code other than "0" are errors that can be omitted because they are not delegations. In other words, their status is that there is a problem (ServFail) or the server refuses(Refused) to answer. The result of this stage is a dataset with SOA records that belong to the delegation endpoint.

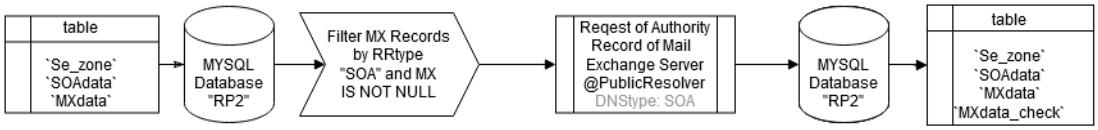

Figure 5: *Process of checking the authority of the delegation*

The author converted the flow from figure 6 into the pseudocode below. As part of the logging, it is important to include a timestamp in the check so that it can be analyzed in the final stage to determine whether the problem is linked to a change at a particular point in time.

IN: `SELECT ‘id‘,‘zonelink‘,‘rcode‘,‘PREFERENCE‘,‘EXCHANGE‘ FROM ‘mxdata‘ WHERE ‘rcode‘ = 0;`
DO: foreach MX record { `dig $EXCHANGE @127.0.0.1 SOA` }
OUT: `INSERT INTO MXdata_check (‘id‘, ‘mxdata‘, ‘zonelink‘, ‘ping‘, ‘pingdata‘, ‘soa‘, ‘whois‘, ‘data‘, ‘date‘)`
`VALUES (’0’, ’21’, ’74’, NULL, NULL, ’example.com.’, NULL, ’example.com.,51,IN,SOA, ns1.example.com.,hostmaster.example.com.,2013010101,1800,900,1209600,300’, ’2020-08-22 22:12:18’)`

3.4 Stage 3: Checking the registration of the delegation

As shown in Figure 6, in the third stage of the methodology involves analyzing the SOA of the delegations collected in the previous stage. When the SOA matches the public suffix-list, the answer is coming from the dns server of the operator of the tld.

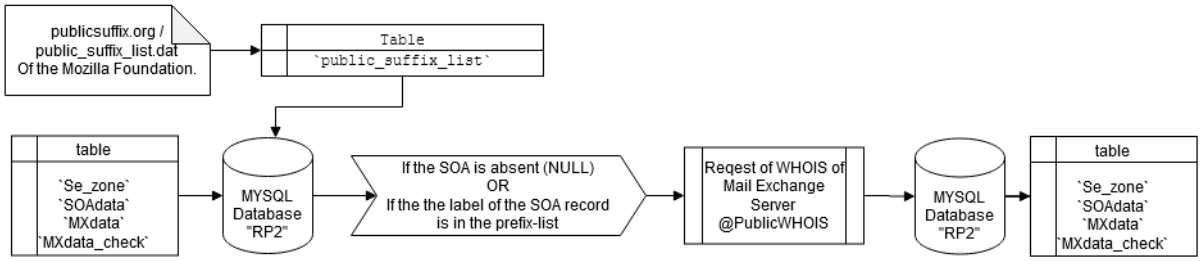

Figure 6: *Method of checking the registration of the delegation*

Mozilla describes a "public suffix" as a domain under which internet users can (or historically could) directly register names. Examples are .se and .nl but also .co.uk,and private initiatives like .co.nl. The list of public-suffix-list[6] a list of those public suffixes as an initiative of Mozilla Foundation.

Finally, the author requested the registrant information of these domains over the WHOIS protocol to have all the available information about them, though were only able to obtain WHOIS information of the domains matching the prefix list because most WHOIS servers have rate-limiting techniques.

3.5 Stage 4: Manually verifying the results

The final stage involved analyzing the result of the previous steps. To make this process easy, the author developed a number of SQL Views. These views are available as an appendix. Based on these results, false positives can be filtered by hand. These false positives included domains like ".google", quarantine domains, and domain names that are restricted in registration.

[6]https://publicsuffix.org/list/public_suffix_list.dat

4 Proof of concept

To test the framework as described in the methodology the author developed a proof of concept(PoC). This chapter discusses the practical setup around the framework and the experiments that were performed to answer the research questions. It also defines the limitations of the research setup and the techniques.

4.1 Database Setup

In order to perform the stages described in the framework the author designed a master-slave setup using a read-write master with multiple read-only slaves. The master server was hosted on the infrastructure provided by the university and the slaves were partly hosted by a cloud provider. The setup is shown in Figure 7.

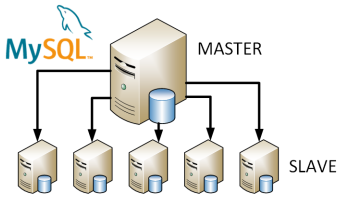

Figure 7: *Master-slave setup.*

The advantage of this setup is that it allows for large scale and independent experiments with faster read times because complex read queries do not cause the whole database to freeze. In addition, this setup assures that not all requests are made with the same source IP address, which is especially advantageous for the WHOIS requests to servers with rate-limiting on a per-IP basis. The disadvantages of the setup are that the slave are not always fully up to date, causing the script to pause until new updates were received, and the master needs to check if the record is unique or else duplicate checks may occur. Figure 8 shows the database structure of the PoC. Moreover, the use of a local DNS cache resolver increases the performance. This study uses Unbound as its DNS cache resolver.

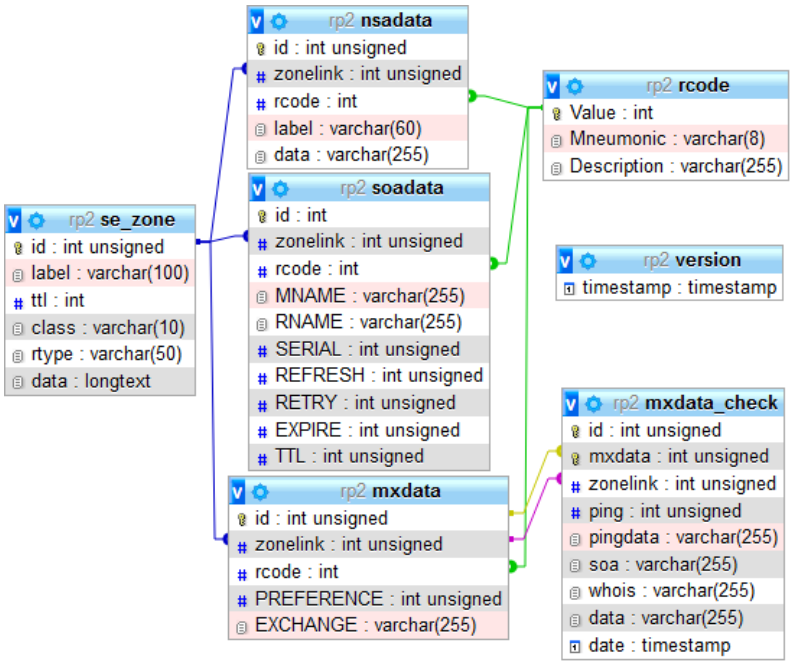

Figure 8: *Entity relation diagram(ERD) of the proposed database*

5 Results

This chapter discusses the results of the experiment. During the project the author developed scripts[7] for the proof of concept. For detailed results, see Figure 10 in the appendix. Figure 9 below shows a limited overview.

	R#	Question	Value
STAGE 0	1	Amount of records in .SE zone at 29th of July	8860606
	2	Amount of NS records in .SE zone at 29th of July	3878855
	3	Amount of Unique Labels (domains) in the .se zone	1471380
	4	Amount of Labels according to Operator	1530949
	5	Amount of NS records per Label in the se-zone	2,636202069
	6	Amount of Domains without NS record in se-zone	59569
		Question	**Value**
STAGE 1	7	Amount of Unique authoritative nameservers in se-zone	53521
	8	Amount of SOA records in at authoritative nameservers	3564890
	9	Amount of MX records in at authoritative nameservers	4844742
	10	Amount of Unique Mailservers (by FQDN)	297898
		Question	**Value**
STAGE 2	11	Amount of Errors on SOA at authoritative nameservers	266120
	12	Amount of Errors on MX records at authoritative nameservers	8215
STAGE 3		**Question**	**Value**
	13	Amount of Domains with Lame NS Delegation (preliminary total)	1106
	14	Amount of Domains with Lame MX Delegation (preliminary total)	2691
		Question	**Value**
STAGE 4	15	Amount of Domains with Lame NS Delegation (Pointing inside)	244
	16	Amount of Domains with Lame MX Delegation (Pointing inside)	2335
	17	Amount of Domains with Lame NS Delegation (Extern)	862
	18	Amount of Domains with Lame MX Delegation (Extern)	356
	19	Amount of Unique Non-existing nameservers	570
	20	Amount of Unique Non-existing mailservers (with active MX pointing)	217
-	21	Most active Top-talker, Non-existing nameserver, with NS pointing	59
-	22	Most active Top-talker, Non-existing mailserver, with MX pointing	52

Figure 9: *Results*

[7]Source code of the scripts is available on request or via https://github.com/bab2501/dnsse

Based on the zone file[8] of the Swedish .se tld, the author determined that there are 1.471.380 active domains with at least one NS record. This figure is consistent with the statistical data published on the operator's website.

The author checked the name servers of the domains below the Swedish .se tld and identified 1.106 NS records as lame delegations. After further investigation, the author found 244 In-Bailiwick[9] nameservers pointing a to child domain and were deleted between the zone transfer and check. The other 862 domains contain lame NS records that point outside of their own domain. These may be vulnerable to takeover.

Within the Swedish .se tld, there are 2.691 domains pointing to 3.631 mail servers on domains that are not registered. However, 3.414 domains with In-Bailiwick mail-servers which point to a child domain and were therefore deleted. Next, 217 mx-servers were identified, which does not exist, but have other .se domains pointing. In total 356 se-domains point to 217 non-existing server.

In the final stage (4), the author checked the list of duplicates by grouping. As a result, we have discovered a few top talkers. The lame nameserver with most incoming lame NS delegations had 59 domains and, the leading lame mail server had 52 delegating domains.

A redacted version of the results can be found in the appendix. Figure 1 shows the top results of lame nameservers and Figure 2 the lame mailservers. In comparing figures, the author noticed that the spread of lame delegations is a fraction greater for nameservers(NS) than for mailservers(MX).

6 Discussion

This chapter discusses the results, though it begins by of the limitations of the research.

- The data stored in the DNS is not static. The experiment has been performed once.
- Changes during the experiment may be falsely identified as lame delegation.
- The research was only performed on the Swedish .se tld.

The results suggest that the domains are pointing to a child domain. These domains were removed after initial zone transfer (stage 0) but before all stages were preformed. The statistics published by the operator also reflect this removal. Figure 16 shows the dates has been added to the appendix.

The results also suggest that most lame delegations are caused by typographical errors, in the delegation such as missing separators. Examples like "mxisp.com." and "mailclusterisp.se.".

This study used the public suffix list. Some false positives among special domains were noted (e.g. .google). These special tld domain names are not available for registration.

[8] Accessed on July 29, 2020.

[9] "In-bailiwick is a modifier to describe a name server whose name is either a subdomain of or (rarely) the same as the origin of the zone that contains the delegation to the name server." - rfc8499 [4]. For instance, the nameserver at ns1.blaauwgeers.se is in-bailiwick for blaauwgeers.se

7 Conclusion

This chapter addresses the research question by answering the sub-questions. The answers are based on the results of the framework developed during the study.

What are lame records and which are possible security implications caused by them? There are several types of lame delegations. According to Pappas, they can be divided into 3 types; non-responding delegation endpoint, malfunctioning endpoints, and non-existing endpoints. This latter type of lame delegations may be vulnerable to takeover if they are open for registration. Takeover delegations can lead to usurpation of control of the service, as in the case of an MX record the email service.

How many lame delegations exist within the .se tld? This research develops a framework for recognizing lame delegations within the Swedish tld. This framework was then tested in a one-off proof of concept during the project.

The author identified 862 domains with Lame NS delegation and identified 356 domains with Lame MX delegations.

Can we identify any top-talkers among them? The author discovered multiple top talkers with more than one domain pointing. The largest unregistered nameserver had 58 delegating domains. The author also identified one unregistered mailserver with 52 delegating domains.

Are the domains below the Swedish .se tld vulnerable to lame delegation takeover? Based on the results of the research, the author conclude that, at the time of the study, a limited number of Swedish domains have lame records to unregistered domains. These domain names are vulnerable to takeover.

7.1 Proposed countermeasures

Based on the conclusion the authors proposes numerous recommendations to prevent the appearance of lame delegations. The proposed measures are intended for administrators of DNS servers.

1) Check the endpoint of the delegation before modification. As a countermeasure, the author proposes that the domain administrator perform an additional check during the modification of DNS delegations of whether there is a running service present at the delegation endpoint that serves the delegated domain. This check can be part of the user interface of the service providers. If no user interface is deployed, the operator can use other techniques like a scheduled script to regularly check the domains that they serve. If there is no service running on the delegation endpoint, a warning should be given to the domain administrator as well as an extra check of whether the change is correct. Many of the lame delegations that the author found are based on typos, which can be prevented by the proposed additional control question.

2) Regularly check the NS delegation on the parent domain As a countermeasure to the operation to the parent zone in this study the operator of the Swedish .se tld should regularly check whether a valid delegation still exists. The operator should also check whether there is an authoritative DNS server available at the registered delegation endpoint, in the NS record. A notification can then be sent to the technical contact email address known to the operator. This check could also be added to the annual audit of the WHOIS contact proposed by ICANN[10].

3) Use of In-Bailiwick Nameservers The final countermeasure, the use of In-Bailiwick nameservers, eliminates by nature the risk of delegation takeover. These in-bailiwick delegations are below the parent. One example is the nameserver for blaauwgeers.se being hosted at ns1.blaauwgeers.se. The disadvantage of this measure is that if the IP address of the nameserver changes, all delegations must be modified by the operator of the parent domain. This can be administratively complex if several domains are managed by a single administrator. This option is particularly useful when one cannot run a control script or if the IP address is not expected to change.

7.2 Future work

Based on the results, this study makes the follow proposals for future work:

- Perform the framework on different TLD's.
- Perform the framework on a regular basis and analyse the difference.
- Analyze the proportion of partially lame delegations, with only some of the records being lame.
- Improve the framework, including third level and other resource record types like SRV, DNAME and CNAME.
- Improve performance of the Proof of Concept (database, code, speed).
- Examine and further develop proposed countermeasures.

[10]https://whois.icann.org/en/basics-whoisfield-section-3

References

[1] Paul Mockapetris. *rfc1034 - Domain names-concepts and facilities*. 1987. URL: https://tools.ietf.org/rfc/rfc1034.txt.

[2] John Naughton. "The evolution of the Internet: from military experiment to General Purpose Technology". In: *Journal of Cyber Policy* 1.1 (2016), pp. 5–28. DOI: 10.1080/23738871.2016.1157619. eprint: https://doi.org/10.1080/23738871.2016.1157619. URL: https://doi.org/10.1080/23738871.2016.1157619.

[3] David Barr. *rfc1912 - Common DNS operational and configuration errors*. 1996. URL: https://tools.ietf.org/rfc/rfc1654.txt.

[4] P Hoffman and A Sullivan. *K. Fujiwara," DNS Terminology*. 2019. URL: https://tools.ietf.org/rfc/rfc8499.txt.

[5] Harald Schioeberg. *DNS Praktikum Protokolldesig*. Technische Universität Berlin. 2008. URL: https://www.net.t-labs.tu-berlin.de/teaching/ws0809/PD_labcourse/PDF/WS2008_protdesign_03-dns-pack-6perPage.pdf.

[6] Amreesh Phokeer, Alain Aina, and David Johnson. "DNS Lame delegations: A case-study of public reverse DNS records in the African Region". In: *International Conference on e-Infrastructure and e-Services for Developing Countries*. Springer. 2016, pp. 232–242.

[7] Vasileios Pappas et al. "Impact of configuration errors on DNS robustness". In: *IEEE Journal on Selected Areas in Communications* 27.3 (2009), pp. 275–290.

[8] Paul Hoffman, Andrew Sullivan, and Kazunori Fujiwara. *DNS terminology*. 2015. URL: https://tools.ietf.org/rfc/rfc7719.txt.

8 Appendices

Source code of the script is available on request or at `https://github.com/bab2501/dnsse`

	R#	Question	Value	Simplified SQL
STAGE 0	1	Amount of records in .SE zone at 29th of July	8860606	SELECT COUNT(*) FROM `se_zone`;
	2	Amount of NS records in .SE zone at 29th of July	3878855	SELECT COUNT(*) FROM `se_zone` WHERE `rtype` = "NS"
	3	Amount of Unique Labels (domains) in the .se zone	1471380	SELECT COUNT(DISTINCT `se_zone`.`label`) FROM `se_zone` WHERE `se_zone`.`rtype` LIKE "NS"
	4	Amount of Labels according to Operator	1530949	*Source (https://internetstiftelsen.se/en/domain-statistics/growth-se/)*
	5	Amount of NS records per Label in the se-zone	2,636202069	= "Result 2" / "Result 3"
	6	Amount of Domains without NS record in se-zone	59569	= "Result 4" - "Result 3"
		Question	**Value**	**Simplified SQL**
STAGE 1	7	Amount of Unique authoritative nameservers in se-zone	53521	SELECT COUNT(DISTINCT `se_zone`.`data`) FROM `se_zone` WHERE `se_zone`.`rtype` LIKE "NS"
	8	Amount of SOA records in at authoritative nameservers	3564890	SELECT COUNT(*) FROM `soadata` WHERE `soadata`.`rcode` = 0 ;
	9	Amount of MX records in at authoritative nameservers	4844742	SELECT COUNT(*) FROM `mxdata` WHERE `mxdata`.`rcode` = 0 ;
	10	Amount of Unique Mailservers (by FQDN)	297898	SELECT COUNT(DISTINCT `mxdata`.`EXCHANGE`) FROM `mxdata` WHERE `mxdata`.`rcode` = 0;
		Question	**Value**	**Simplified SQL**
STAGE 2	11	Amount of Errors on SOA at authoritative nameservers	266120	SELECT COUNT(*) FROM `soadata` WHERE `soadata`.`rcode` <> 0
	12	Amount of Errors on MX records at authoritative nameservers	8215	SELECT COUNT(*) FROM `mxdata` WHERE `mxdata`.`rcode` <> 0
		Question	**Value**	**Simplified SQL**
STAGE 3	13	Amount of Domains with Lame NS Delegation (preliminary total)	1106	SELECT COUNT(DISTINCT `label`) FROM VIEW 'LameNS'
	14	Amount of Domains with Lame MX Delegation (preliminary total)	2691	SELECT COUNT(DISTINCT `label`) FROM VIEW 'LameMX'
		Question	**Value**	**Simplified SQL**
STAGE 4	15	Amount of Domains with Lame NS Delegation (Pointing inside)	244	WHERE `label` LIKE "%.NAMESERVER" (In-bailiwick namesevers)
	16	Amount of Domains with Lame MX Delegation (Pointing inside)	2335	WHERE `label` LIKE "%.EXCHANGE" (In-bailiwick mailsevers)
	17	Amount of Domains with Lame NS Delegation (Extern)	862	WHERE `label` NOT LIKE "%.NAMESERVER" (External namesevers)
	18	Amount of Domains with Lame MX Delegation (Extern)	356	WHERE `label` NOT LIKE "%.EXCHANGE" (External mailsevers)
	19	Amount of Unique Non-existing nameservers	570	SELECT COUNT(DISTINCT `NAMESERVER`) FROM VIEW `LameNS` GROUP BY `NAMESERVER`
	20	Amount of Unique Non-existing mailservers (with active MX pointing)	217	SELECT COUNT(DISTINCT `EXCHANGE`) FROM VIEW `LameMX` GROUP BY `EXCHANGE`
-	21	Most active Top-talker, Non-existing nameserver, with NS pointing	59	SELECT COUNT(DISTINCT `label`) FROM VIEW 'LameNS' SORT BY `label` (....)
-	22	Most active Top-talker, Non-existing mailserver, with MX pointing	52	SELECT COUNT(DISTINCT `label`) FROM VIEW 'LameMX' SORT BY `label` (....)

Figure 10: *Detailed Results*

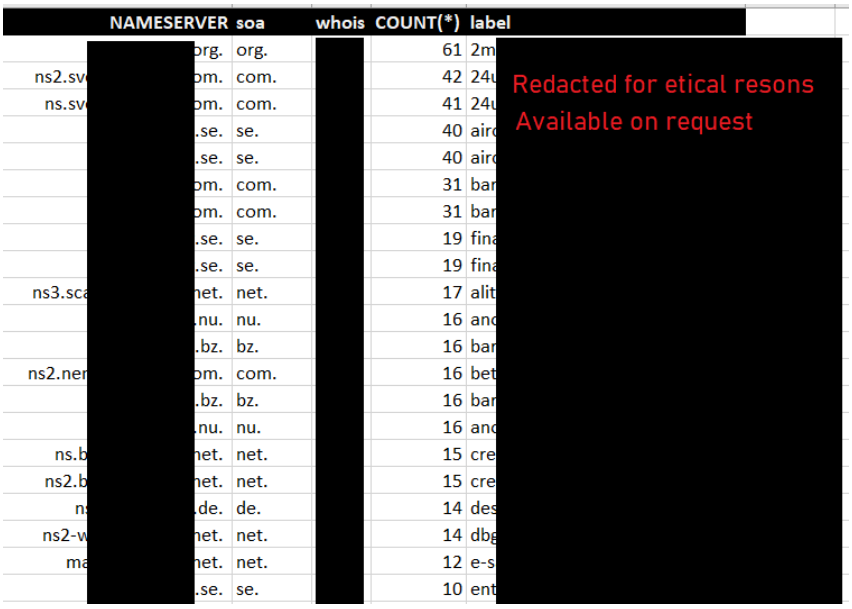

Figure 11: *Redacted version of the results on NS lame delegation.*

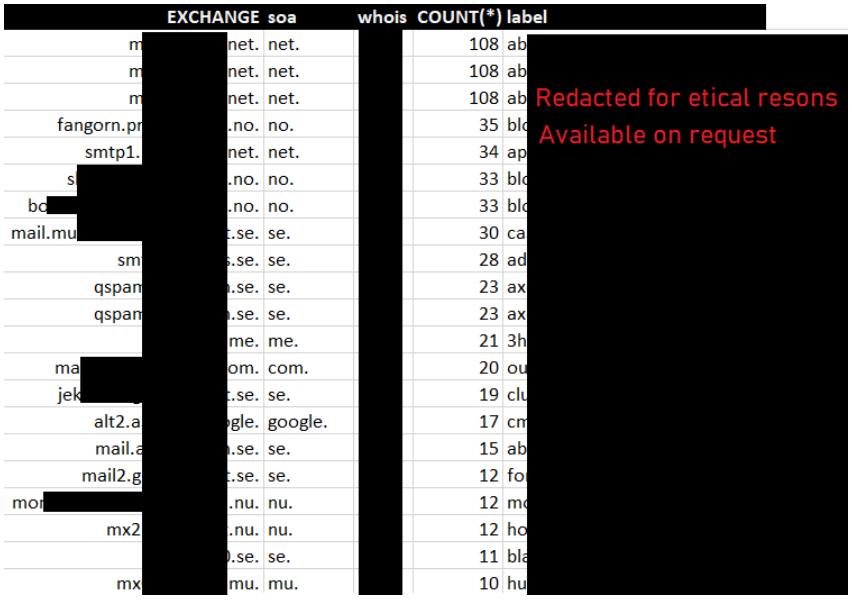

Figure 12: *Redacted version of the results on MX lame delegation.*

LameNS.sql

```
CREATE VIEW 'LameNS' AS SELECT
        `se_zone`.`data` AS "NAMESERVER",
        GROUP_CONCAT(DISTINCT `nsdata_check`.`soa`) AS "soa",
        GROUP_CONCAT(DISTINCT `nsdata_check`.`whois`) AS "whois",
        COUNT(*),
        GROUP_CONCAT(DISTINCT `se_zone`.`label`) AS "label"
FROM `se_zone`
RIGHT JOIN `nsdata_check` ON `nsdata_check`.`zonelink` = `se_zone`.`id`
WHERE `nsdata_check`.`soa` IN (SELECT `prefix` FROM public_suffix_list)
AND `se_zone`.`rtype` = "NS"
GROUP BY `se_zone`.`data`
ORDER BY COUNT(`se_zone`.`data`) DESC;
```

Figure 13: *SQL View NS*

LameMX.sql

```
CREATE VIEW 'LameMX' AS SELECT
        `mxdata`.`EXCHANGE` AS "EXCHANGE",
        GROUP_CONCAT(DISTINCT `mxdata_check`.`soa`) AS "soa",
        GROUP_CONCAT(DISTINCT `mxdata_check`.`whois`) AS "whois",
        COUNT(*),
        GROUP_CONCAT(DISTINCT `se_zone`.`label`) AS "label "
FROM `mxdata3`
RIGHT JOIN `mxdata_check` ON `mxdata_check`.`mxdata` = `mxdata3`.`id`
RIGHT JOIN `se_zone` ON  `mxdata_check`.`zonelink` = `se_zone`.`id`
WHERE `mxdata_check`.`soa` IN (SELECT `prefix` FROM public_suffix_list)
GROUP BY `mxdata`.`EXCHANGE`
ORDER BY COUNT(`mxdata`.`EXCHANGE`) DESC;
```

Figure 14: *SQL View MX*

results.sql

```sql
CREATE VIEW `results` AS
SELECT
(SELECT COUNT(*) FROM `se_zone`) AS `m1`,
(SELECT COUNT(*) FROM `se_zone` WHERE `rtype` = "NS") AS `m2`,
(SELECT COUNT(DISTINCT `se_zone`.`label`) FROM `se_zone` WHERE `se_zone`.`rtype` LIKE "NS") AS `m3`,
"1530949" AS `m4`,
"C3" / "C4" AS `m5`,
"C5" - "C4" AS `m6`,
(SELECT COUNT(DISTINCT `se_zone`.`data`) FROM `se_zone` WHERE `se_zone`.`rtype` LIKE "NS") AS `m7`,
(SELECT COUNT(*) FROM `soadata2` WHERE `soadata2`.`rcode` = 0) AS `m8`,
(SELECT COUNT(*) FROM `mxdata3` WHERE `mxdata3`.`rcode` = 0) AS `m9`,
(SELECT COUNT(DISTINCT `mxdata3`.`EXCHANGE`) FROM `mxdata3` WHERE `mxdata3`.`rcode` = 0) AS `m10`,
(SELECT COUNT(*) FROM `soadata2` WHERE `soadata2`.`rcode` <> 0) AS `m11`,
(SELECT COUNT(*) FROM `mxdata3` WHERE `mxdata3`.`rcode` <> 0) AS `m12` ,
(SELECT COUNT(DISTINCT `label`) FROM `LameNS`) AS `m13`,
(SELECT COUNT(DISTINCT `label`) FROM `LameMX`) AS `m14`;
```

Figure 15: *SQL View Results*

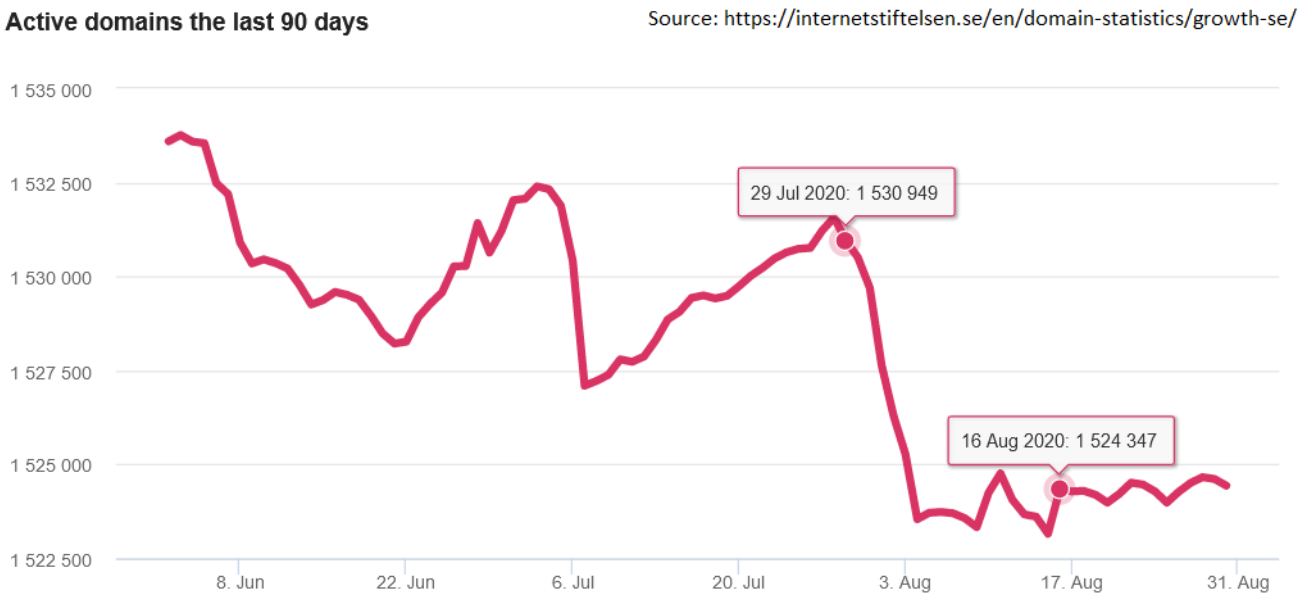

Figure 16: *SE statistics*