Measuring the cost of adding security on Domain Name System (DNS)

Daniel Migault, Security Lab, Francetelecom R&D {mglt.biz@gmail.com}

Abstract— **Domain Name System (DNS) is a public infrastructure that makes Internet so easy to use. On the other hand this protocol has been so popular that other protocols such as ENUM, or other private databases are thinking of implementing DNS-like database. 3G, IMS are other examples of wireless architecture that relies on DNS, especially for routing purposes, and the discovery of SIP-proxies.. In a near future, it seems then that DNS-like databases will host many kind of information, some being confidential, some needing dynamically being updated...**

Whereas public or private, DNS until now, was not really implementing much security. This paper aims at measuring the cost and impact of different security extensions on the Domain Name System (DNS), such as DNSSEC, TSIG, as well as other security protocol such as IPsec. Tests were performed both on single servers as well as on a network environment.

By doing so, we think that anyone relying on a DNS architecture public or private will have the key element to secure this architecture, according to its needs, such as dynamic update, scalability, time response, CPU load...

***Index Terms*— DNS, DNSSEC, Naming architecture, Security**

I. INTRODUCTION

Internet mainly relies on DNS system which enables the binding between a human readable domain names such as www.francetelecom.com and an IP address. This naming system was elaborated when security was not considered as a critical issue. Then, a few security extensions were added to the DNS protocol (DNSSEC rfc2535 [5], rfc4033 [2], rfc4034 [4], rfc4035 [3], TSIG rfc2845 [9], SIG(0) rfc2931 [6]). Nevertheless those options are still not fully deployed over the global Internet, so today DNS can't be considered as secure.

On the security point of view, frequent attacks against DNS servers, show DNS's Internet Achill's heel position. DNSSEC deployment is more and more required. However, this protocol standardized in March 2005 implies radical changes on both the structure and the architecture of the current DNS. Indeed DNSSEC is using heavy security mechanisms, that impact DNS protocol.

DNS system is mainly used to enable naming service, binding a domain name and an IP address. However, new protocols such as ENUM for example, gives DNS new perspectives of services, and so new way to handle DNS databases, with new security requirements as confidentiality, authentication of clients...

This article aims at measuring costs generated by the use of DNS security extensions. This article will present results of a bench of tests of various DNS security extensions (DNSSEC and TSIG), as well as other security protocols such as IPsec (rfc2401[8], [10]). Tests were all performed on the same DNS architecture, so that comparison between those security extensions can easily be done. Once security cost has been highlighted, it is possible to deploy a secure DNS database. Selection of relevant security mechanisms is based on the specific security requirements, as well as performances the data base should meet, taking into account the global network environment. On the other hand, by knowing the cost of security extensions, administrators could also plan a migration scenario by first modifying the architecture, and then deploy required security extensions.

II. TESTED DNS ARCHITECTURE DESCRIPTION

Security tests on DNS were performed on the DNS architecture presented in figure 1 and figure 2. The DNS architecture is considering two zones :

- A "father zone" whose domain name is "rootdomain"
- A son zone whose domain name is "subdomain.rootdomain"

Network architecture of this platform is composed of three DNS servers:

- One hosting the rootdomain zone file
- Two DNS servers hosting the subdomain zone subdomain.rootdomain. One of the servers is a master DNS server, and the other is the slave DNS server

The tested DNS architecture platform is connected the IPv6 experimental network [13] of France Telecom R&D. The DNS cache-forwarder server is resolving requests sent by the clients.

Servers are Pentium II and III at 500 Mhz with 128 MB memory, running OS Linux Ubuntu (kernel 2.6). DNS servers software is BIND 9.3.1 ([1]). Requests were sent using the client of this distribution, as well as a home developed DNS Java client.

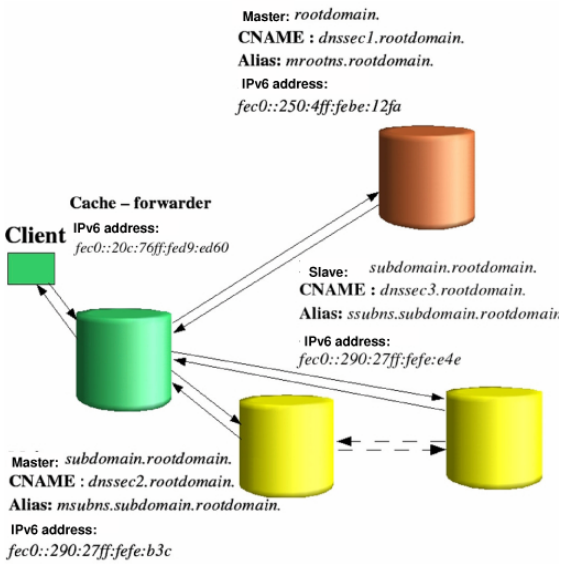

Figure 1: Tested DNS/DNSSEC architecture

III. GOALS AND STRATEGY

A. *DNS Security mechanisms description*

DNSSEC stands as DNS security extension, and has been standardized by the IETF in documents rfc4033 [2], rfc4034 [4] and rfc4035 [3]. This extension enables data integrity check, while packets go through the network from server to client as well as DNS data source authentication. Data source authentication is possible trough the establishment of a chain of trust between the DNS servers. The chain of trust enables to go from one trusted DNS servers to trusted subdomain DNS servers. Thus DNSSEC enables you to get a response from a DNS server you can trust. Trust is based on cryptographic public / private key properties.

TSIG and SIG(0) (DNS Request and Transaction SIGnature) are DNS extensions that enable authentication.

TSIG ((rfc2845 [9]) is signing data using symmetric cryptography. Client and DNS servers authenticate themselves by sharing a private key. Since communicating entities are using pre-shared keys, authentication between nodes is nor scalable. Thus TSIG can only be used with a very few number of nodes. TKEY (rfc2930 [7]) is a mechanism that enable setting up a TSIG key between different nodes.

SIG(0) (rfc2931 [6]) is using asymmetric cryptography. It enables a DNS server to encrypt a DNS transaction with a private key. The public key is on the server and available in the zone file. Whereas DNSSEC is signing data, with the private key, SIG(0) is signing the exchange.

IPsec (rfc2401 [8]) (Internet Protocol Security) enables confidentiality, and authentication issues of IP datagrams. IPsec is not dealing with data, but with data flows exchanged between two IP nodes. So IPsec is authenticating the source IP address of the device that is sending DNS data rather than the authority that has generated the DNS data. On the other hand, DNSSEC is authenticating the source of the DNS data. One can have access to those DNS data, from requesting a cache server, a primary server, or any kind of device, the signature will be the same, since the same key is used to sign DNS data. Using IPsec, authentication is based on a specific IP address, and so DNS requests have to be sent to that specific server. IPsec, in our configuration, is using symmetric cryptography, which, compared to SIG(0), needs less resources. Furthermore, IKE (Internet Key Exchange) is a protocol that enables management of SA (Security Association) between peers.

B. *Differences between security mechanisms*

To point out differences between mentioned above security mechanisms, some aspects might be considered:

Which way data are encrypted? Encryption can be either symmetric or asymmetric. Symmetric encryption requires that both entities exchanging data own a shared secret key. The main advantage of using symmetric encryption mode is that it costs far less than asymmetric encryption. In fact, for a given security level, far less calculation resources are required when symmetric encryption are considered. On the other hand one of its drawbacks is that mechanisms to set in a secure way, the shared secret key between the nodes must be provided. Furthermore, since both entities are using the same secret key, authentication cannot be proceeded as with asymmetric cryptography. One can easily understand that if two nodes and only two nodes are sharing a key, by decrypting data, one node can deduce data has been sent from the other node. In that sense one can think of authentication with shared key. This is not anymore the case when the key is shared by more than two nodes! By sharing the key between more than two nodes, encrypted data might have been emitted from any node that owns the key. As a result, one can't prove data has been emitted from one or the other node.

What kind of security is implemented? Security is a generic concept and can be considered from different ways, implying different security mechanisms or security properties. Security properties considered are the one mentioned below:

- **Confidentiality:** this property enables data sent through a network to be read and understood only by specific users. Encryption is usually used for confidentiality purpose. Most of the time encryption operations require algorithm, and keys. Whoever doesn't have the appropriated key will not be likely to have the clear and understandable data. Theses who are owning the appropriate key will need to decrypt the encrypted data. The decryption process also requires keys. When symmetric cryptography is used, the key can be a pre-shared key which means that data is encrypted and decrypted with the same key. When asymmetric cryptography is used, a pair of public / private key is required. Data encrypted with the private key will be decrypted thanks to the public key, and data encrypted with the public key will be decrypted with the private key.
- **Integrity:** this property enables a user to assert the received data has not been changed by a third entity. Hash functions are often used in that purpose. A hash function is a one-way function that takes input data of any size and generates an output data of a given size. In our purpose, a hash function is such that if two data are different,

hashes are different in term of probability. This enables one to check whether or not a given data has been changed from the time it has been sent and the time it has been received.

- **Authentication:** this property enables the node that receives the data to check the identity of the sender. When public / private keys are involved, a data encrypted with a private key can only be decrypted with the public key. By using this process, a node that is owning a pair of public / private key can send data encrypted with the private part of the public / private pair of keys to another node. Only nodes that are owning the public part of the sender node's public / private pair of keys, will be likely to decrypt the data. Since such data could only have been sent by the node that is owning the private key, the sender is then authenticated. Also data can be the meaningful data, but if one is only interested in using asymmetric cryptography for authentication, the encrypted data can be only the hash of the meaningful data. By doing so the encrypted data is far less smaller. This process is called the signature process. As we saw in previous section, authentication can also be considered with pre-shared keys, as long as the key is shared between two, and no more than two nodes.

What is the encrypted data? Considering one encryption method, one needs to know what to encrypt. Although IPsec and DNSSEC are not using the same encryption method, our purpose in that subsection is not to discuss about used encryption methods. Our purpose is to point out the nature of the encrypted data, and its meaning in term of security for network deployment.

In the case of IPsec, IP datagram between two nodes are encrypted. This means that security is implemented at the transport layer, and enables to secure communication between nodes. On the other hand, DNSSEC implement security at the application layer, by encrypting the data itself, and providing signature.

The difference is then the same as if one was considering a letter with a signature. We do generally don't care about who puts the letter into the mailbox. We are more concerned about who signed the letter, and use the signature at the bottom of the letter to authenticate the person who wrote the letter. This is DNSSEC point of view. By appending a signature to the data at the application
layer, DNSSEC client doesn't care about who provides the data to the client. DNSSEC client are rather concerned about who signed the data, and if the nodes that signed the data is a trusted node.
If one goes on with this analogy, another way to consider security is to care about who provides the data, and consider that data is reliable when coming from a trusted node. This is IPsec's point of view. IPsec, is securing the transport layer, and so relies rather on nodes then on the data.

Those two visions have an impact in term of network deployment. IPsec aims at securing a tunnel between two nodes, each IP datagram will be encrypted on the flow. This means that pre-calculation is not possible. In term of network deployment the use of IPsec means the trust is based on connectivity to trusted nodes. This would then require that clients connect themselves directly to the trusted node. So this would mean that all traffic will be concentrated on the trusted nodes, which would probably be in our case the authority servers. On the other hand, since DNSSEC is signing data at the application layer, data can be sent to the client by non trusted nodes such as relay servers. This is called the cache mechanism. Cache servers are collecting answer from previous requests, and answer to coming request with the answer they are storing. This mechanism enables DNS / DNSSEC traffic to be split between cache servers and authority servers. Note that this mechanism is possible because security is related to the data, and signatures are sent with the corresponding data. Asymmetric cryptography is used, which costs more than asymmetric cryptography, but that, in other hand enables pre-calculation. In fact communicated data are known in advanced, and do not vary from one client to the other. In that sense signatures do not need to be generated on the flow, and can be pre-computed.

C. Goals of the tests

The tests presented in this paper aims at giving keys to people that are looking to secure their DNS infrastructure. Cost of security will be presented considering different criteria such as functionalities, performances, as well as different configurations.
The main configuration taken into account are :

- **DNS:** This configuration is the reference we are using to measure the security cost of other configurations.
- **DNSSEC:** This configuration is mainly concerned by DNS data integrity check and authentication. This DNSSEC security extension is not symmetric in the sense that only data from the server are concerned by security operations. The client is not authenticated by the server for example, and no security operations on client requests are being performed by the server.
- **DNS + IPsec:** This configuration is rather used in server-to-server communication or client-to-server when the number of clients is relatively small. In fact, IPsec tunnel requires configuration which can be a problem to scalability, when the number of client increases. On the other hand the main advantage of this configuration is to add confidentiality.
- **DNSSEC + IPsec:** This configuration is used to secure links between servers with IPsec, in order to secure all transactions, and not only DNS data. DNSSEC is in that sense considering the communication between clients and servers.
- **TSIG :** This configuration is only use with server to server communication. In fact TSIG is using symmetric cryptography and so both communicating entity need to be configured. On the contrary to IPsec that can use IKE for key negotiation, there is no key negotiation protocols specifically designed for TSIG communication. At last, on the contrary to IPsec, TSIG is not

implementing confidentiality of transferred data, but is only focused on integrity purpose.

D. Tests Strategy

For each mentioned configuration, this paper aims at measuring, in a quantified way, the impact of security on the DNS architecture. This impact with be measured thanks to the following criteria:

- Time response to requests
- CPU load generated by the DNS process
- Updating time

Tests are made in two distinct environments :

- Single server
- Simulating DNS network architecture

Tests on a single server aim at measuring the security impact on specific devices. Those kind of tests will be called server test. Nevertheless, DNS servers are part of a network, and specific servers test might generates border effect on the network or on the Naming architecture. Measurement of security cost on the global naming architecture will be called Network tests.

Tests can be of three distinct natures:

- **Unitary Tests:** Those tests aim at measuring server and network behavior on a per request base. In order to get readable results, tests need to be performed a certain number of time to meet probability requirements.
- **Load Tests:** Those tests aim at measuring server and network behavior in a network environment, i.e. when more than one client are sending requests. In that purpose, many parallel clients will be simulated.
- **Update Tests:** Those tests aims at measuring the servers / network capacity to deal with updates. In this paper, tests will be proceeded in the following order.

At first unitary test will be made, then, loading tests. At last, updating tests will be performed. Tests concerning time response and memory / CPU load, two types of DNS client were used: the traditional DNS client dig, that is part of the BIND 9.3.1 ([1]) distribution, and a Java home-made client specifically developed for the tests purposes. If dig client is waiting for the answer of a request before proceeding to the next request, the java client is working in a asynchronous way. It uses two threads, one that is sending requests, and the other that is receiving responses and that analyses them. By its design it is more adapted for variable loading tests.

Updating tests are using the traditional nsupdate of BIND 9.3.1 distribution. To generate requests automatically, bash scripts are used.

IV. UNITARY TESTS

A. Unitary test description

The first test aims at evaluating impact on single DNS request performed by dig client. In that purpose we do care not to overload neither the servers nor the network. dig client is sending request N + 1 only once request N has been answered. Requests are of same type in any tested configuration. DNS servers are using CPU time and memory. Those data are given through system command regarding the DNS process also referenced as named.

B. Security cost and time response

Figure 2 shows time response expressed in millisecond (ms) depending on the configuration. The third line is measuring security cost in percentage (%) by comparing time responses of configuration that implements security and time response of the DNS only configuration.

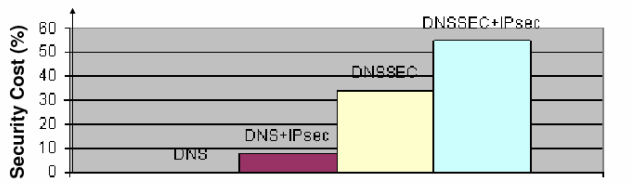

Figure 2 : Unitary tests on different security configurations

Figure 2 clearly shows impact of the different security mechanisms on time response. If ”A < B”means that B is costing more than A, by considering time response we have the following statement :

DNS < (DNS+IPsec) < DNSSEC < (DNSSEC + IPsec)

DNSSEC costs at lot higher then IPsec because of the use of asymmetric cryptography, which cost more then symmetric cryptography. Furthermore, the client not only have to decrypt, but also to check signatures, which means that it needs to compute a hash of the received data, then decrypt the SIG type data sent by the server, and compare it with the computed Hash. In fact SIG type data on DNSSEC server are DNS data signatures, that is to say hash of DNS data encrypted with the private part of the server pair of keys. All of these make DNSSEC time response much higher then IPsec time response.

The cost of the (DNSSEC + IPsec) configuration is higher then the sum of the cost of DNSSEC configuration and the cost of the IPsec configuration. In fact DNSSEC data are more verbose, so IPsec, is, in this configuration, encrypting more datagram than in the IPsec only configuration.

C. Security cost and time CPU

The table below shows security cost of the different configurations according to time CPU of the DNS (named) process. The cache-forwarder server is resolving requests in a recursive way. This means that a request sent by the dig client to the cache forwarder generates a first request from the cache-forwarder to the rootdomain DNS server, followed by a second request from the cache-forwarder to the subdomain DNS server. Of course servers are configured to work in a iterative way, which means that the answer from server to the cache-forwaders are only made of DNS data that are within servers’ zone file. In this tests all servers devices are the same.

In addition to decryption operation, a request sent by the client generates two requests from the cache-forwarder. One is sent to the rootdomain DNS serve, and one is sent to the subdomain DNS servers. Since there is load balancing between master and slave DNS server, one out of two is sent to the slave subdomain DNS serve and one out of two is sent to the master subdomain DNS server.
Time CPU is quite low because the dig client is sending a request only after it has received the response of the previous request. This shows that network latency is much more important than DNS resolution load, which gives DNS / DNSSEC server a high availability.

Tested server /Config	DNS	(DNSSEC+IPsec)	DNSSEC	(DNSSEC+IPsec)
dnssec1 rootdomain server	0.82 % [0%]	1.21 % [+47%]	1.25% [+52%]	2.89% [+259%]
dnssec2 (master subdomain server)	0.20% [+0%]	0.36% [+80%]	0.30% [+50%]	0.90% [+350%]
dnssec3 (slave subdomain server)	0.20% [0%]	0.35% [+75%]	0.30% [+50%]	0.73% [+265%]
Cache forwarder	7.20% [+0%]	7.40% [+2%]	8.80% [+22%]	8.80% [+22%]

Figure 3 : Security cost and time CPU

Analysis of the results in figure 3 requires to consider DNS servers according to there type. For the rootdomain DNS server, one can consider security cost as below:

DNS << (DNS + IPsec) < DNSSEC <<< (DNSSEC + IPsec)

One can notice the very small difference between time CPU required for DNSSEC and time CPU required with IPsec. This clearly shows the balance between sending more data, and encrypting data. As mentioned in previous section, signatures are pre-computed in DNSSEC, so that DNSSEC server doesn't have to proceed to any encryption operation. On the other hand DNSSEC servers need to deal with much bigger zone files than traditional DNS server, and DNSSEC generates much more traffic than DNS (SIG and NSEC types are only parts of DNSSEC protocol, and they are carrying quite huge amount of data, compared to traditional DNS data). IPsec encrypt smaller data, but each communication requires encryption operation. CPU time is equal by using either DNSSEC or IPsec, means that encryption operation for DNS data costs as much as sending specific DNSSEC types. From zone file analysis, one can assert that a DNSSEC response requires 14 times more bandwidth than a DNS response.
For the subdomain master and slave DNS server, one can consider the cost of security depending on the different configuration as below :

DNS < DNSSEC < (DNS + IPsec) << (DNSSEC + IPsec)

Difference between the CPU time required in the DNSSEC and (DNS + IPsec) configuration is rather small. In this specific case, IPsec tunnels have also been configured between master and slave. The difference is then even smaller then in the case of the rootdomain server.
Difference between time CPU required for the subdomain servers, and the rootdomain server is due to response size difference and load balancing. If one considers the DNS protocol, the rootdomain server is sending an NS type, that indicates the name of the next server to send the request to, and a A type, which gives the IP address of the server. If one considers DNSSEC, one should add, the NSEC and SIG associated types. The subdomain servers are only sending one A type data. So rootdomain answers are twice bigger. Furthermore, traffic on subdomain servers is load balanced between the master and the slave. In term or bandwith, a single subdomain server is dealing with 1/4 of the traffic of the rootdomain server.
An analysis between cache-forwarder performances and other servers is quite hard since we are using different devices in this test. Nevertheless the security cost according to time CPU is as below :

DNS < IPsec << DNSSEC < (DNSSEC + IPsec)

The main difference between this server and the others, is that this one is dealing with much more traffic than the other servers. In fact, one can notice that IPsec, generates on this server, a higher cost than the on the other devices (in percentage). This shows that loading tests are necessary to evaluate cost of security.
Although loading tests are required for a complete security cost evaluation, but as far as unitary tests are concerned, it happens that a classification of security cost regarding to time CPU is as below :

DNS < (DNS + IPsec) < DNSSEC << (DNSSEC + IPsec).

D. Security Cost according to memory load

Figure 4 presents the percentage of used memory on the different servers, according to the different network configuration.

Tested server /Config	DNS	(DNSSEC+IPsec)	DNSSEC	(DNSSEC+IPsec)
dnssec1 rootdomain server	1.60 % 	1.60%	1.60%	1.60%
dnssec2 (master subdomain server)	1.70%	1.70%	1.70%	1.80%
dnssec3 (slave subdomain server)	1.70%	1.70%	1.80%	1.80%
Cache forwarder	0.40%	0.40%	0.40%	0.40%

Figure 4 : Security cost and memory load

Memory load is proportional to the zone file size. BIND DNS servers are loading the whole zone file into their cache memory. If memory is not large enough, the server crashes. Further tests shows up that the size of the zone file has no incidence on time response. In our tests, little zone file were used so used memory on the DNS server is not large.
Time to live (TTL) of DNS data is set to zero, so the cache forwarder doesn't need to keep all already requested data, and so it memory is not increasing during the tests. Nevertheless more memory is required when DNSSEC is used. In fact DNS cache forwarder behaves as a client and so needs to keep contexts of quite large data.

E. Conclusion

This first set of tests shows the cost of security on unitary requests using different network security configurations. IPsec and the use of symmetric cryptography make this protocol better than DNSSEC when time response and time

CPU are considered. Nevertheless the differences go down when load is increasing.
As far as memory load is concerned, it happens that DNSSEC requires more memory than IPsec because of heavier datagram and computing operations.
In all tests involving IPsec, IKE key negotiation was not considered. We only considered pre-established tunnels. The client used in unitary test was dig part of the BIND 9.3.1 distribution. This client has not been designed for loading tests purposes. This is why a new client has been designed for our loading tests.

V. LOADING TESTS

Since now the DNS client used to perform test is not anymore the dig client, but a home made client. This home made java client enables us to adjust the rate of sending requests as well as analyse answers contend. It then enables us to simulate simultaneous DNS client among the network, in a ”real” environment.
This section focus especially on overload behaviour and on properly answered requests rate. Those parameters will be tested on a single server (single server tests) or on the global architecture (network tests).

A. Time CPU and correct responses – single server test

Those tests tend to show the highest acceptable rate of requests for the servers, and so to measure the influence on performances as well as behaviours of those different servers, according to the different security configuration.

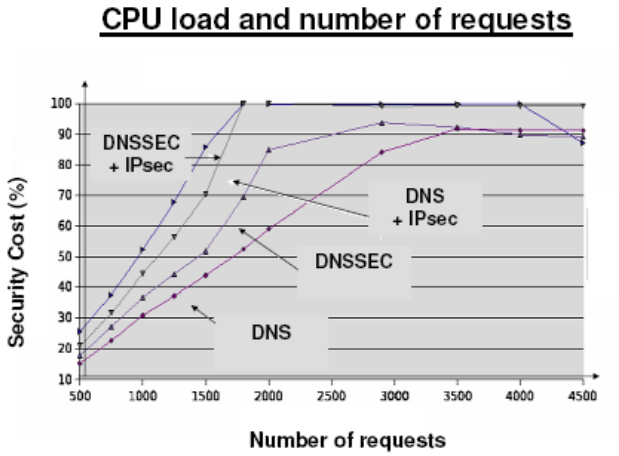

Figure 5: CPU load according to number of sent requests, with different DNS configurations,

For each configuration one can notice two breaking points :

- The server breaking point: Any configuration has a maximum CPU load that a server can accept. The server breaking point is the lowest rate of sent request where the maximal CPU load is reached. Figure 4 illustrates this breaking point by considering the CPU load regarding to the rate of sent requests. It shows that with IPsec, the maximal CPU load can be of 100% whereas with no IPsec configuration, the CPU load is around 90%. This shows that an overloaded device using IPsec, can eventually be unreachable, not only for DNS requests but for any other operations.
- The service breaking point: Whereas the server breaking point is defined with the maximum CPU load of the device, the service breaking point is rather looking to properly answered requests. The service breaking point is then defined as the maximum rate of sending request a DNS server can answer properly. Whereas server breaking point is looking at server overload, the service breaking point is looking at service overload.

Regarding the CPU load with the different configuration, it happens that the security cost is as mentioned below:

DNS < DNSSEC < (DNS+IPsec) < (DNSSEC + IPsec)

On the contrary to unitary tests, DNSSEC seems to present advantages over IPsec in loading tests. This is easily explained by the fact that sending all DNSSEC information costs more then signing a few requests, but when the number of requests increases, it becomes easier to send data then to encrypt data. Implied processes don’t manage contexts the same way.

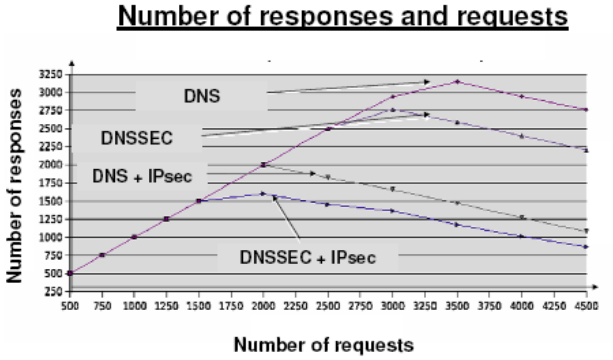

Figure 6: Number of correct answers according to number of sent requests with different

Nevertheless before the server is not able to answer anymore to request, there is a intermediary point where the DNS service is not conducted properly, and requests might be answered, but not with a proper answer. This means that a request whose answer is contained in the zone file get an answer with an error message. This point is called the service breaking point. The figure shows those service breaking points for different configurations. Tests are performed with direct connection from client to tested server, and the cache-forwarder server is not considered.
Figure 4 shows the maximum CPU load of the different security configurations, which are 100% or 90%. Overload point is considered for CPU load over 90% of the maximum CPU load, that is to say 90% for the first and 81% for the other.

Configuration	DNS	(DNS+IPsec)	DNSSEC	(DNSSEC+IPsec)
90% load of server breaking point	3000	2000	1800	1500
Security cost	0%	+33%	+40%	+50%

Figure 7: Overload CPU point with the different security configuration

Figure 8 then makes it possible to evaluate in terms of requests the maximum loading supported by DNS servers according to various configurations. This table confirms the higher cost during loading tests of IPsec compared to DNSSEC.

Figure 6 highlights that for small values of the load, the number of answers is equal to the number of requests for all the configurations. For larger values, the graph is not linear any more. The service breaking point corresponds to the maximum rate of requests that the server is able to treat correctly. The position of the critical point depends on the configuration and is summarized in Figure 7.

Configuration	DNS	DNSSEC	(DNS+IPsec)	(DNSSEC+IPsec)
service breaking point	3000	2500	2000	1500
Security cost	0%	+17%	+33%	+50%

Figure 8: Maximum treatment capacity of naming servers using different security configuration

Various security configurations evaluation shows that protocol DNSSEC is more robust than IPsec in network overload situation. Thus, DNSSEC seems to cost twice less than IPsec in term of a maximum number of treated DNS requests.

B. Security cost according to CPU load on the network architecture

It is still a loading test, but this time influence on the network architecture is considered. The java developed client sends requests to the network architecture through the cache forwarder server as an intermediary point or a relay.
During this test, the relay happens to be a bottleneck. For a reduced rate of requests, the resolution process is properly managed with almost 100 % of correct answers. When rate increases, the resolution process fails from time to time and either answers of type SERVFAIL are sent to the client, or requests are directly discarded by the server and doesn't even answer to the requests. The client now makes the difference among the received answers between the correct answers with a NO ERROR tag and the error messages with a SERVFAIL tag. For the tests involving DNSSEC, there are two possibilities: the customer can decide to ask the relay not to check signature validity or to make it (default option).
The first option can be specified by putting the bit CD (Checking Disable) at 1 in the request. Instead of four types of tests to be made, now there are six of them because of six possible combinations.

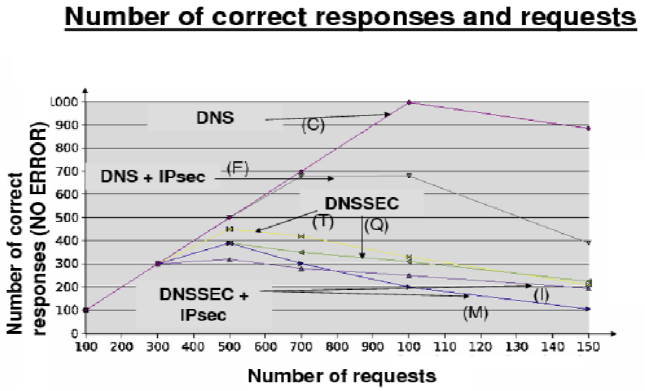

Figure 9: Number of properly answered requests regarding to the number of sent requests, using different security configuration, as well as the signature check option. Configurations are : DNS (C), (DNS + IPsec) (F), (DNSSEC + IPsec) with signature check performed by cache forwarder server(I), (DNSSEC + IPsec) with no signature check performed by the cache forwarder (M), DNSSEC with signature check performed by the cache forwarder server (Q), and DNSSEC without signature check performed by the cache forwarder (T).

Just like in the former test, the figure Numbers properly answered requests regarding to the number of sent requests (figure 9) shows a breaking point for each different configuration tested. Breaking points are visible in figure 10.

Config	DNS	DNS + IPsec	DNSSEC (sig check)	DNSSEC (no sig check)	DNSSEC + IPsec (sig check)	DNSSEC + IPsec (no sig check)
Max numb. of properly answered req.	1000	700	400	450	300	400
Security cost	0%	+30%	+60%	+55%	+70%	+60%

Figure 10: Network architecture capacity to deal with traffic load and answer properly.

Regarding to the ratio of the number of properly answer request / number of sent requests, the security cost is as below :

DNS < (DNS+IPsec) < (DNSSEC without checks) < (DNSSSEC with checks) < (DNSSEC without checks + IPsec) < (DNSSEC with checks + IPsec)

Values reported in the table are, at first glance quite surprising compared to results get in the single server case.

The fall of performances when considering the whole network architecture is very important if protocol DNS is made secure. The number of properly answered requests is much smaller than the values we get in the single server bench of tests. In fact the cache-forwarder server is dealing with the entire resolution which includes sending requests to the proper server, opening contexts, analyzing responses, checking signatures... So the cache-forwarder happens to be quickly overloaded, much faster then previous servers which were mainly answering to requests in a iterative way.

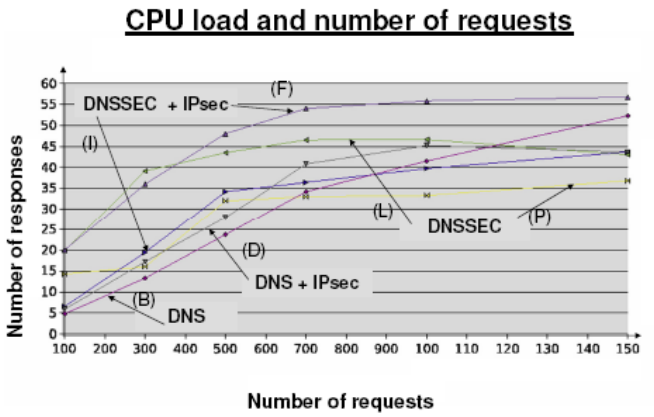

Figure 11: CPU load regarding to the number of requests, the different security configuration, and activation of the check signature option. Different configurations are the following : DNS (B), (DNS + IPsec) (D), (DNSSEC + IPsec) with signature check option activated (F), (DNSSEC + IPsec) without signature check option activated (I), DNSSEC with signature check option activated (L), and DNSSEC without signature check option activated (P).

Considering and analysing figure 7 as we did in previous test, it becomes clear that the maximum number of requests answered correctly in each configuration is the cache-forwarder service breaking point. This test highlights the key role of the cache-forwarder DNS server in the architecture, and its bottleneck position. Concentrating all requests the cache-forwarder server needs to be properly designed.

VI. Conclusion

These tests show that security has a considerable impact on the DNS server / network architecture performances. Parameters for both single servers and global network architecture such as: the time response, the maximum capacity of request treatment and the update time are to be taken into account when secure DNS deployment is considered.

There is no security extension that is recommended rather then the other.

Since there is no one-best-security-extension, security extensions considered for deployment have to be chosen according to the service requirements. In that sense, TSIG proved to be rather inexpensive during our update tests and seems well adapted to make safe transactions between servers. Quite easy to configure, TSIG is mainly focused on integrity, and authentication. Manual configuration on communicating nodes makes TSIG not scalable. IPsec on the other hand is also focused on confidentiality and provide a key negotiation mechanism (IKE), which makes it scalable. Once the tunnel is established, security cost is similar to TSIG's cost. DNSSEC on the other hand, does not seem to fit architecture that requires frequent updates. When time response is considered on an not-loaded architecture, (DNS + IPsec) seems better than DNSSEC. On the other hand, when robustness to heavy load is considered DNSSEC seems to be better than (DNS + IPsec). In fact DNSSEC certify the information whereas IPsec and TSIG provides authentication of the node one is communicating with.

This property is a great advantage for balancing load thought the network.

These tests were considering on cost of securing the binding between a domain name and an IP address. To go even further and to secure the bind between an IP address and an Ethernet address, SEND seems to be a interesting point to look at.

VII. Aknowledgment

I would like to thank very much Bogdan Marinoiu for all testing results and Jean Michel Combes for supporting our investigation inn DNSSEC.

References

[1]. Paul Albitz and Cricket Liu. Dns et bind, 2002. Edition : 4.

[2]. R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), March 2005.

[3]. R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. RFC 4035 (Proposed Standard), March 2005.

[4]. R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. RFC 4034 (Proposed Standard), March 2005.

[5]. D. Eastlake 3rd. Domain Name System Security Extensions. RFC 2535 (Proposed Standard), March 1999. Obsoleted by RFCs 4033, 4034, 4035, updated by RFCs 2931, 3007, 3008, 3090, 3226, 3445, 3597, 3655, 3658, 3755, 3757, 3845.

[6]. D. Eastlake 3rd. DNS Request and Transaction Signatures (SIG(0)s). RFC 2931 (Proposed Standard), September 2000.

[7]. D. Eastlake 3rd. Secret Key Establishment for DNS (TKEY RR). RFC 2930 (Proposed Standard), September 2000.

[8]. S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401 (Proposed Standard), November 1998. Updated by RFC 3168.

[9]. P. Vixie, O. Gudmundsson, D. Eastlake 3rd, and B. Wellington. Secret Key Transaction Authentication for DNS (TSIG). RFC 2845 (Proposed Standard), May 2000. Updated by RFC 3645.

[10]. P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136 (Proposed Standard), April 1997. Updated by RFCs 3007, 4033, 4034, 4035.