The Secure Domain Name System pattern

Eduardo B. Fernandez[1] and Michael Van Hilst [2]

[1]Dept. of Computer Science and Engineering, Florida Atlantic University, USA, ed@cse.fau.edu
[2] Nova Southeastern University, Fort Lauderdale, FL, USA, vanhilst@alum.mit.com

Abstract
The Domain Name System (DNS) is a hierarchical distributed naming system for resources connected to the Internet. It associates addresses with domain names assigned to each of the participating entities. We present here the Secure DNS pattern, which includes protection against availability and integrity attacks and can authenticate the origin of messages.

Keywords: Domain Name System, Security patterns, Internet security, cloud computing

Introduction

The Domain Name System (DNS) is a hierarchical distributed naming system for resources connected to the Internet. It associates addresses (system names) with domain names (user names) assigned to the resources. Because of its importance, it has become a potential target for terrorists, and several attacks have already happened, e.g., the Syrian Electronic Army, a pro-Assad group, altered the DNS records used by the New York Times, Twitter, and the Huffington Post [Rag13].

We present here a Secure DNS pattern, incorporating security patterns for its protection. This is an ideal version of the current DNS servers used in the Internet where the need for backward compatibility has resulted in a complex implementation. We describe the new pattern using a modified POSA template [Bus96] and our intended audience includes web and cloud architects, system designers, and application developers. This pattern will be added to a catalog of security patterns [Fer13]. It has, of course, value on its own.

Secure Domain Name System (SDNS)

Intent

The Secure Domain Name System (SDNS) is a hierarchical distributed naming system for any resource connected to the Internet. It associates network addresses with domain names assigned to the participating entities. It includes protection against availability and integrity attacks and can authenticate the origin of messages.

Example

A cloud service provider did not have any protection for its DNS and it became the target of numerous attacks against its customers. Its reputation suffered significantly and lost many customers.

Context

Web-based computing systems and other distributed systems using the Internet for communication.

The DNS infrastructure is made up of computing and communication entities that are geographically distributed throughout the world. There are more than 250 *top-level domains*, such as .org and .com, and several million *second-level domains*, such as nsf.gov and ietf.org. Accordingly, there are many name servers in the DNS infrastructure, with each containing information about a small portion of the domain name space. The domain name data provided by DNS is intended to be available to any computer located anywhere in the Internet. There are no well-defined system boundaries—participating entities are not subject to geographic or topologic confinement rules. Every domain has a Domain Authority. Users buy domain names from a domain *registrar*. The registrar publishes lookup information for the domain it has registered with a Master Domain Authority for the domain zone of the domain and stores the Resource Record in its database.

Problem

The DNS is a fundamental element in the Internet and a great target for attackers of all kinds because of its importance and its lack of protection. Because DNS data is meant to be public, preserving the confidentiality of DNS data pertaining to publicly accessible IT resources is not a concern [Cha3]. The primary security goals for DNS are data integrity and source authentication, which are needed to ensure the authenticity of domain name information and to maintain the integrity of domain name information in transit [DNSS, Rag13] Availability of DNS services and data is also very important; DNS components are often subjected to denial-of-service attacks intended to disrupt access to the resources whose domain names are handled by the attacked DNS components. Its mapping data can also be attacked.

Because of these characteristics, conventional network-level attacks such as masquerading and message tampering, as well as violations of the integrity of the hosted and disseminated data, have a completely different set of functional impacts [Cha13], as follows:

- An impostor that spoofs the identity of a DNS node can deny access to services for the set of Internet resources for which the node is supposed to provide information (i.e., domains served by the node).

- Domain information spoofing can redirect all incoming traffic for a domain to a server of the attacker's choosing. This enables her to launch additional attacks, or collect traffic logs that contain sensitive information [Rag13].

- The attacker can capture all inbound email for a domain [Rag13]. This option also allows the attacker to send email on their behalf, using the victim organization's domain and take advantage of their reputation.

- Bogus DNS information provided by an attacker can poison the information cache of other DNS nodes providing that subset of DNS information, resulting in a denial of service to the resources serviced by it.

- Violation of the integrity of DNS information resident on its authoritative source or the information cache of an intermediary that has accumulated information from several historical queries may break the chained information retrieval process of DNS. This could result in either a denial of service for the DNS name resolution function or misdirection of users to a harmful set of resources.

- If the name resolution data (data used to find other data) hosted by the DNS system violates content requirements as defined in DNS standards, it could have adverse impacts such as an increased workload on the DNS system, or serving obsolete data that could result in denial of service to Internet resources. In most software, program data independence (as in conventional Database Management Systems (DBMS)) provides a degree of buffering against adverse impacts due to erroneous data. In the case of DNS, the data content determines the integrity of the entire system.

The solution to this problem is constrained by the following forces:

***Defenses to known attacks*—**We need to control all the identified attacks.

Defense method independence—We should be able to tailor the type or level of security if there are new attacks.

Tailoring—We need to adjust the policies to fit different customers.

Rule maintenance—It should be easy to adapt to new security policies.

Compatibility—The introduction of new policies should not adversely affect the use of services which are not yet supporting those new policies.

***Scalability*—**If the number of customers increases we will have a consequent increase in the number of rules which will make the situation even more confusing and may lead to errors by the administrators. We should avoid this situation.

Solution

NIST recommends [Cha13]:

- Implement appropriate system and network security controls for securing the DNS hosting environment, such as operating system and application patching, process isolation, and network fault tolerance.

- Protect DNS transactions such as update of DNS name resolution data and data replication that involve DNS nodes within an enterprise's control. The transactions should be protected using hash-based message authentication codes based on shared secrets.

- Protect query/response transactions that could involve any DNS node in the global Internet using digital signatures based on asymmetric cryptography, as outlined in IETF's Domain Name System Security Extension (DNSSEC) specification.

The solution in this pattern uses a combination of the security defenses of the IETF Secure Inter-Domain Routing Working Group, based on a Resource PKI approach for authentication (DNSSEC) and cloud providers who use ACLs to protect DNSs in their systems.

DNSSEC uses PKI digital signatures. The correct DNSKEY record is authenticated via a chain of trust starting with a set of verified public keys for the DNS root zone. Each DNS domain authority has a key signing key signed by a DNS domain authority higher in the DNS chain. Domain owners generate their own keys, and upload them using their DNS control panel at their domain-name registrar, which in turn pushes the keys via DNS to the zone operator (e.g.: Verisign for .com) who signs them with its domain key signing key, and publishes them in DNS.

Structure

Figure 1 shows the class diagram of this pattern. The **DNS** manages **Name Resolution Data**, which is protected from modification using **Role-Based Control** (the data is public for reading). Two types of transactions can be performed. **Query Transactions** require **Digital Signature** protection while **DNS Transactions** require **Hash-Based Authentication**. **Users** can perform any of these transactions.

Dynamics

Figure 2 shows the message sequence in the Use Case corresponding to a DNS Query Transaction. A user sends a URL to the DNS which translates it to an IP address. This address is signed for authentication and returned to the user.

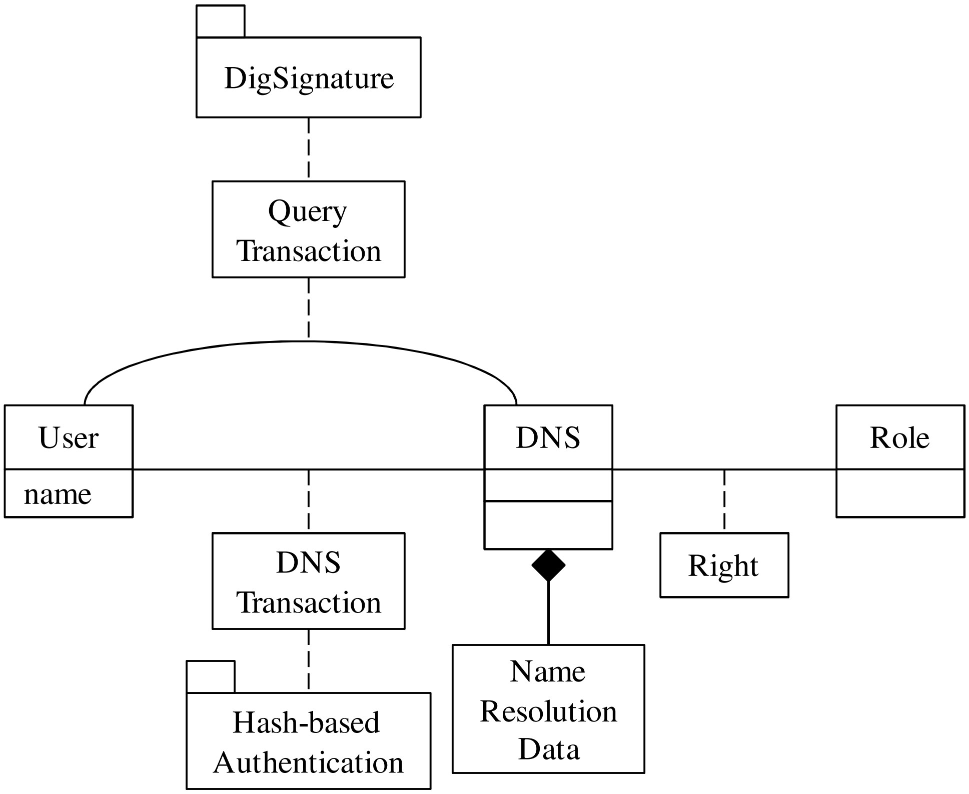

Figure 1. Class diagram of the SDNS pattern

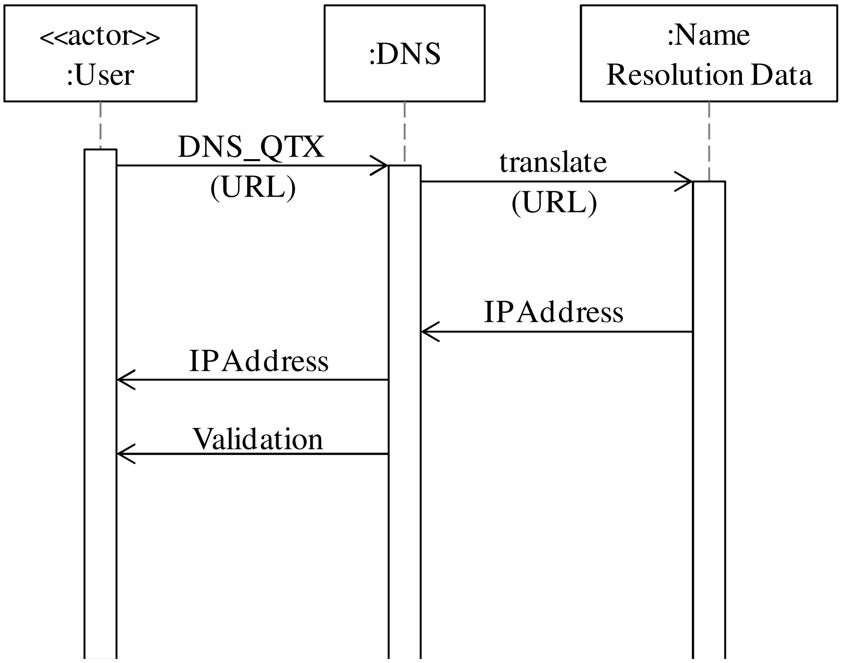

Figure 2. Sequence diagram for the use case “DNS Transaction”

Implementation

The **Domain Name System Security Extensions** (**DNSSEC**) is a suite of IETF specifications for securing certain kinds of information provided by the DNS as used on IP networks [DNSS, Dom]. It is a set of extensions to DNS which provide to DNS clients (resolvers) origin authentication of DNS data, authenticated denial of existence, DNS data integrity, and means for public key distribution, but not availability or confidentiality.

Protecting IP addresses is the immediate concern for many users, but DNSSEC can protect any data published in the DNS, including text records (TXT) and mail exchange records (MX), and can be used to bootstrap other security systems that publish references to cryptographic certificates stored in the DNS.

DNSSEC responses are authenticated but not encrypted. DNSSEC *does not* protect against DoS attacks directly, though it indirectly provides some benefit (because signature checking allows the use of alternate potentially untrustworthy parties).

The data in DNS is organized in a tree structure with a designated authoritative service for each node (zone) in the tree (but not the leaves). The authoritative service is easily identified, as is the chain of authentication going back to the root zone, for their keys. The public key used to verify the signature is available in a DNSKEY record for that authoritative zone service. Keys in a chain are authenticated by finding a DS record of the key from its authoritative parent, signed by the private key of that parent. In this way, the chain can be followed back to the Root (or to a

"Trusted Anchor" higher in the tree which is already known and verified by the resolver.). This operation can be recursive or using stubs.

To simplify key update, key signing keys and zone signing keys are separate. The authoritative service signs its own zone signing keys with its key signing key. In that way, it can update its zone key and signatures more frequently without the involvement of its parent. During a transition period, pending parent and trusted anchor updates, two Signatures may be presented for old and new keys.

Known uses
Early adopters of DNSSEC include Brazil (.br), Bulgaria (.bg), Czech Republic (.cz), Puerto Rico (.pr) and Sweden (.se), who use DNSSEC for their country code top-level domains. Several ISPs have started to deploy DNSSEC-validating DNS recursive resolvers. Comcast became the first major ISP to do so in the United States, completing deployment on January 11, 2012.

VMWare [VMw11] and Rackspace [Rac] are using ACLs in their cloud DNSs products.

It is possible to use a discretionary access control list (DACL) in Windows Server 2012 to control the permissions for the Active Directory users and groups that may control the DNS zones [Mic12]

Example resolved
The cloud SP is now using a SDNS and the attacks mentioned earlier cannot happen, except perhaps DDoS, which can only be mitigated.

Consequences
Some advantages are:

***Defenses to attacks*—**We have defenses for all the identified attacks, which can prevent or mitigate most of them.

Defense method independence—We could use different signature or hashing approaches or even other type of defenses because the security mechanisms are separated from the DNS main functions.

***Tailoring*—**We can tailor the access control rules to fit precisely the needs of each customer.

Rule maintenance—To adapt to new policies we only need to add or modify some rules.

Compatibility—The introduction of new policies does not affect the use of services which are not yet supporting those new policies.

***Scalability*—**If the number of customers increases we can just increase the Name Resolution Data or add new levels of indexing.

Liabilities include:

- Reduces speed and requires costly changes to the systems that use the current approach.
- The need to design a backward-compatible standard.
- Deployment of implementations across a wide variety of DNS servers and resolvers (clients)
- Overcoming the perceived complexity of SDNS and its deployment
- It does not protect information about which records are being sought by the client (no privacy protection).

Related patterns

Digital Signature with Hashing [Has09].

Role-Based Access Control (RBAC) [Fer13].--Describe how to assign rights based on the functions or tasks of people in an environment in which control of access to computing resources is required and where there is a large number of users, information types, or a large variety of resources.

The paper [Ate01] discusses a specific aspect of DNS.

Conclusions

This pattern describes how to secure a fundamental unit of the Internet. Like all patterns, its validation will happen when designers use it in their systems. Several countries and vendors are already using this reinforced DNS. As indicated, this is a rather abstract Secure DNS but that is the objective of most patterns.

Acknowledgements

We thank our shepherd Jose Francisco Ruiz for his comments that improved our paper.

References

[Ate01] Giuseppe Ateniese and Stefan Mangard, *Proceedings of the 8th ACM conference on Computer and Communications Security (CCS'01),*86-95, ACM 2001, doi>10.1145/501983.501996

[Bus96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerland, and M.Stal, *Pattern- oriented software architecture*, Wiley 1996.

[Cha13] R. Chandramouli and S. Rose, *Secure Domain Name System (DNS) Deployment Guide*, NIST Special Publication 800-81-2, Sept. 2013

[DNSS] http://en.wikipedia.org/wiki/DNS_spoofing

[Dom] Domain Name System Security Extensions, http://en.wikipedia.org/wiki/Domain_Name_System_Security_Extensions

[Fer13] E. B. Fernandez, *Security patterns in practice - Designing secure architectures using software patterns*. Wiley Series on Software Design Patterns, Wiley 2013.

[Has09] K. Hashizume, E.B.Fernandez, and S. Huang, "Digital Signature with Hashing and XML Signature patterns", *Procs. of the 14th European Conf. on Pattern Languages of Programs, EuroPLoP 2009.*

[Mic12] Microsoft Windows Server 2012, "Controlling DNS zones". http://technet.microsoft.com/en-us/library/cc755193.aspx

[Rac] http://www.rackspace.com/cloud/dns/?cm_mmc=PPC_NonBrand-_-Search_Non_Brand_-_LATAM_-_English%3EDNSGeneral%3EGeneral%3EBroad-_-Match-_-secure%20dns&gclid=CN3RkYjarr4CFcNj7AodjFwAkg

[Rag13] S. Ragan, Three types of DNS attacks and how to deal with them, 2013 http://www.csoonline.com/article/2133916/malware-cybercrime/three-types-of-dns-attacks-and-how-to-deal-with-them.html

[Vix] P. Vixie, "What DNS is not", *CACM*, (52)12, 43-47

[WMw11] VMware, "Securing the cloud—A review of cloud computing, security implications and best practices", VMware Corp., 2011