The Effect of Diversity on the Security of the DNS

BENOIT BAUDRY, AZAR HOSSEINI
Department of Software and Computer System
KTH Royal Institute of Technology
Electrum 229

This proposal is a good reason to investigate the various attacks on DNS protocol and study the effects of diversity to control units of a DNS resolver. During this document I try to answer some important questions about disadvantages of software monoculture and lead my proposal to provide a proper platform against DNS attacks. This desire will be accessible while we can calculate the probability of attack occurrence.

1. WHY MONOCULTURE IS PROBLEM?

Nowadays security is considered as a multi-aspects that can ensure users to save their information, protect their connection and have all other positive points in internet surfing. One of the momentous concepts is software monoculture that despite its advantages in lower security cost, being more compatible and its ability to adopt the popular product, it can amplify online threats. This claim arises from common features that can cause the same disease. For example, when I and my neighbor use the same door-lock then both of us are vulnerable to the same theft. To sum up the problems of software monoculture we can refer to the: BOBE[1] attacks [1], similar risks exist in network computer systems and getting a limited amount of diversity [2].

2. HOW TO REDUCE THE PROBLEM OF MONOCULTURE?

The more common a product is, the more it will suffer from infection by malware, therefore, a more diverse product would better resist infection. “Scott Charney, chief security strategist for Microsoft] says monoculture theory doesn’t suggest any reasonable solutions; more use of the Linux-source operating system, a rival to Microsoft Windows, might create a ‘duoculture,’ but that would hardly deter sophisticated hackers. True diversity, Charney said, would require thousands of different operating systems”. Thousands of different OSs may help to increase the diversity number but it can also reduce security because of its heterogeneous components that are vulnerable to the wide spread of attacks. I suggest that the diversity number should be limited to our ability to control system. Aside from this ability, if we want to be on the internet we should use the limited standards and protocols like TCP/IP, HTML, PDF and all other sorts that are used by everyone else. This claim assumes that we are not allowed to fly to a land with thousands of operating systems.

[1] break once, break everywhere

2018

3. WHAT IS THE BENEFITS OF GAME THEORY?

According to the previous section when we encounter different attacks we should control our system and switch to the same operation but in the other way like other codes, units or process. This controlling operation needs to predict the kind of attacks which can be done by powerful detectors like IDSs and choose the best codes, processes and units which have not prior vulnerability. This switching is based on probabilities and game theory that plays a key role in controlling operation. In chess, we always start with d4[2] but depends on black moving we can choose c4 or e3 then cc3 or cf3 or fd3. This game gets more complicated and has a lot to do with the choices of both sides just like selecting for switching in control unit. In diversity we would like to understand the strategies of attackers and defenders and analyze how much choice is needed to protect system [3].

3.1 Diversity Measures in Games

Saran Neti in [3] assumed bipartite graph by representing a set of n hosts, m vulnerabilities, k mappings from every host to a subset of V(vulnerability), deg(vi) as a degree of common vulnerability among hosts or software then calculate the diversity number of N_a and Renyi Entropy ($\log(N_a)$). This idea can help us to expose our solution under the umbrella of probabilities at next section.

4. WHAT DO YOU WANT TO DO AS A SOLUTION?

This section points to some of the solutions presented in previous documents and is also an introduction to my own solution.

4.1 Previous Solutions

Network consists of different aspects that cause different solutions like:

Aspects [4]	Effects [4]	Solutions
Users	Different needs, ages, habits, computer experience	Different choices
Operating Systems	Unexpected problems, options for implementations	Different layers
Display	Different toolkits, command-lines, texts	Different boxes
Hardware	Different memory, power, servers, systems	Different speed

[2] Each square of the chessboard is identified by a unique coordinate pair—a letter and a number. The vertical columns of squares, called files, are labeled a through h from White's left (the queenside) to right (the kingside). The horizontal rows of squares, called ranks, are numbered 1 to 8 starting from White's side of the board. d4 refers to the position of white pawn and cc3 refers to the square c3 which filled by white knight.

Modularity	Separating into reusable pieces, Considering testability	Provides a topical solution to the problem
Programming languages	Different ways	Easier switching because of the coordination between languages

A summary of the solutions presented in the articles is as follows:

1- Using multi-tier [1]:
Different configuration blocks besides different data centers, servers and libraries.

2- Using code transformer [4]

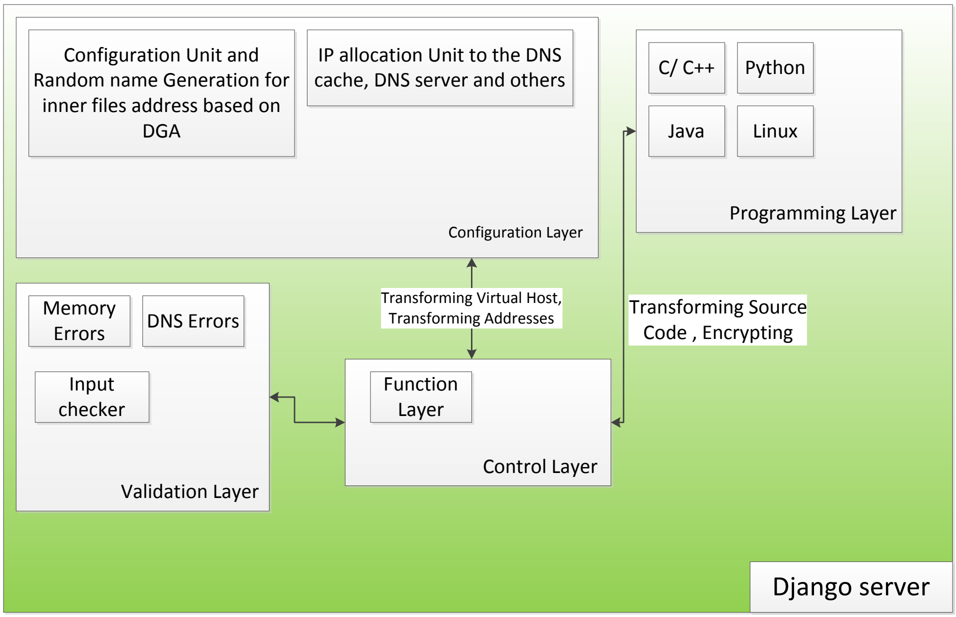
3- Modularity [5]:
Separate inner parts of server (DNS caches from DNS servers).
This solution help us save incoming DNS data from outgoing DNS data and conversely.

4- Using check threat list [6]:
Quad9 prevent the computers and IoT devices from browsing into malicious domains: it checks the site against IBM X-Force threat intelligence that includes 800+ terabytes of threat intelligence data

4.2 Proposed Solution

If we seek a secure DNS platform we should design an integrated system to monitor all layers and all boxes and has a control unit for switching while malicious detector sends its alarms to it. Based on [3] we assume a large graph to connect different units with different roles in our system. The main purpose of this system focuses on DNS protocol.

1- Using Django as a DNS Server:
This type of server is free from the default definitions and can be configured in accordance with the user's preferences. For example we can alter the name address of /etc/resolve.conf to the other intended name for preventing zone files from attacker.


Configuration Unit and Random name Generation for inner files address based on DGA
IP allocation Unit to the DNS cache, DNS server and others
Configuration Layer
C/ C++
Python
Java
Linux
Programming Layer
Transforming Virtual Host, Transforming Addresses
Transforming Source Code , Encrypting
Memory Errors
DNS Errors
Input checker
Validation Layer
Function Layer
Control Layer
Django server

2- Diverse Options:

Layers (L)	**Tasks (T)**	**Inputs (I)**	**Vulnerabilities (V)**	**Global Distribution (G)**	**Attacks (A)**
Layer 1	Task 1	Input 1	Vulnerability 1	GD 1	Attack 1
Layer 2	Task 2	Input 2	Vulnerability 2	GD 2	Attack 2
…	…	…	…	…	…
Layer n	Task m	Input i	Vulnerability j	GD w	Attack q

3- Relationships Between Options:

In previous document I talked about various aims for a secure DNS Resolver as follows:

1) Offloading DNS resolution to separate database
2) Dropping TCP/UDP Packets filtering
3) Avoiding man-in-the-middle exploitation
4) Controlling options like EDNS0, RES_USE_EDNS0 or RES_USE_DNSSEC
5) Avoiding "Resolver Redirection Attacks" and "DNS rebinding attack"
6) Detecting "Data Exfiltration" and "Advanced Persistent Attacks"

This proposal lets me divide these aims to organized parts that can be controlled faster and easier and finally accelerates the process of Diversity Number Calculation.

Here we have six options: Layers (L), Tasks (T), Inputs (I), Vulnerabilities (V), Global Distribution (GD) and Attacks (A) that are related to each other severely.

For explaining more please let me to illustrate details of each option:

Layers:

- Configuration Layer:

  Tasks:

  - Allocating IPs and separating caches from other units:

    (For Offloading DNS resolution to separate database)

    (Using private database and internal firewall to Avoiding "Resolver Redirection Attacks" and "DNS rebinding attack")
  - Random name generating
  - Transforming addresses
  - Transforming Virtual host

  Inputs:

  Vulnerabilities:

  Attacks:

  Global Distribution:

  - Resolver Redirection Attack
  - DNS Rebinding Attack

- Programming Layer

  Tasks:

  - Transforming source code

    (Using related control code in wrapper code to avoid man-in-the-middle exploitation)
  - Encrypting

Inputs:
Vulnerabilities:
Attacks:
Global Distribution:

- Validation Layer

Tasks:

  - Detecting Errors (memory, DNS protocol)
  - Checking Inputs:
    (Dropping TCP/UDP Packets filtering)
    (Detecting "Data Exfiltration" and "Advanced Persistent Attacks")

Inputs:
Vulnerabilities:
Attacks:
Global Distribution:

- Control Layer

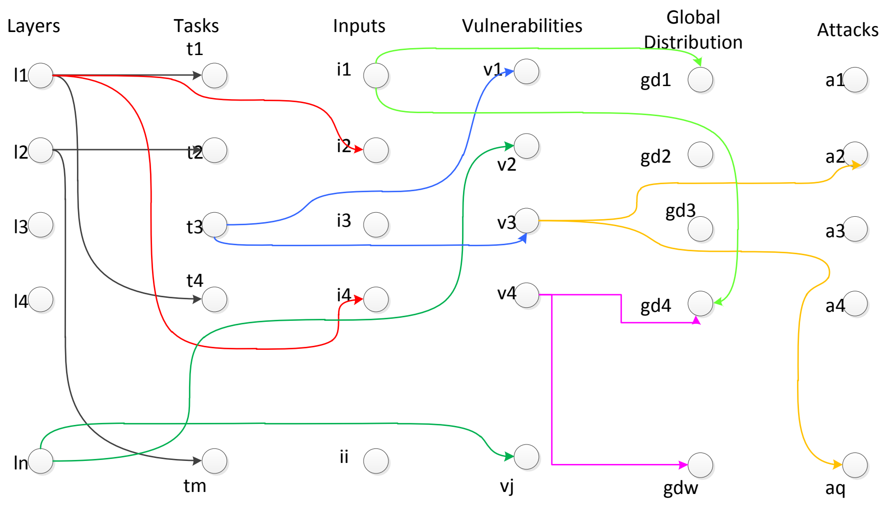
Tasks:

  - Choosing the best option and declaring the command (Switching)
  - Monitoring and Improving Functionality:
    (Controlling options like EDNS0, RES_USE_EDNS0 or RES_USE_DNSSEC)

Inputs:
Vulnerabilities:
Attacks:
Global Distribution:

After filling all options we can calculate Diversity Number accurately. But in this stage we can consider each option as a black box with numerous T, I, V and A.

4- Diversity Number:



If we suppose that Layers are related to all others, Tasks are related to Vulnerabilities, Inputs are related to Global Distributions and Vulnerabilities are related to Attacks and Global Distributions then we have below equations:

1- $\deg(l_n)|_T$, $\deg(l_n)|_I$, $\deg(l_n)|_V$, $\deg(l_n)|_{GD}$, $\deg(l_n)|_A$ # $\deg(l_n)|_T$ describes the degree of l_n based on number of connections between l_n and vertices of T.
2- $\deg(t_m)|_L$, $\deg(t_m)|_V$
3- $\deg(i_i)|_L$, $\deg(i_i)|_{GD}$
4- $\deg(v_j)|_L$, $\deg(v_j)|_{GD}$, $\deg(v_j)|_A$
5- $\deg(a_q)|_L$, $\deg(a_q)|_V$
6-

$$p_{a_q} = \sum_{k=1}^{q}\sum_{r=1}^{n}\frac{\deg(a_k)}{\deg(l_r)N} + \sum_{k=1}^{q}\sum_{r=1}^{j}\frac{\deg(a_k)}{\deg(v_r)J}; N = 1+2+..+n; J = 1+2+..+j$$

This equation tells the possibility of the qth attack on each layer with its related vulnerabilities.

5. HOW IS YOUR SOLUTION GOOD?

One of the benefits of the proposed method is that it can be switched between different parts easily. Another advantage of this method is to calculate the probability of an attack occurrence and strengthen controls of associated part.

6. CONCLUSIONS

This proposal is a successful paradigm to explain precisely the worth of software diversity against software monoculture. During the sections of this proposal we familiar with various diversities, different solutions and their effects on secure system to detect attacks and scape by switching to the other way.

REFERENCES

1. Simon Allier and et.al, “Multi-tier diversification in web-based software application”, *IEEE Software,* Vol.32, 2015, pp.83-90.
2. Bruce schneier, “the dangers of a software monoculture”, Meeting cloud computing compliance mandates, 2010.
3. Saran Neti and et.al, “Software diversity: Security, Entropy and Game Theory”, *HotSec'12 Proceedings of the 7th USENIX conference on Hot Topics in Security*, 2012.
4. James E.Just and Mark Cornwell, “Review and Analysis of Synthetic Diversity for Breaking Monoculture”, *WORM '04 Proceedings of the 2004 ACM workshop on Rapid malcode.*
5. *https://cr.yp.to/djbdns/separation.html*

6. *https://www.ibm.com/developerworks/community/blogs/9c59f17b-ed09-474a-87ac-e2f45ae9eb01/entry/Quad9_configure_your_DNS_server_and_stay_safe?lang=en*

7.