Impact of Diversity on Security of DNS Resolver based on Game Theory

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. INTRODUCTION

Nowadays, security of information allocates the most proportion of essential issues. By increasing the variety of attacks on different services (FTP server, DNS server, WEB server,..) the detection methods have been soared. For detecting intrusion and evaluate the amount of detriment and then designing the prevention scheme, we should know that what is our viewpoint about attacks and how do we want to do it. First and foremost step to do it, is dividing systems into two different kinds, a large system with various parts or a simple system. A large system with various parts can imply diversity feature and a simple system can be monoculture software. This step is a good reason to open the new window to the difference between monoculture and diversity. Both of aspects have their own merits that will be explained in the next section. The second step is to determine that what our vulnerabilities are and what the risk of each one is. The libraries of known vulnerabilities as CVEs are accessible in the internet and we can match them to our system. Collecting the vulnerabilities help us to prioritize the attacks. The third step is related to gather all threats of our system and the information of their reciprocal attacks. The forth step is assuming the comprehensive scheme or algorithm to detect the most proportion of attacks. For this step we suggest to obey one of the game theories to predict attacker's decision and calculate the probability of percentage of her success. The result of this calculation will help control unit to make the best decision for detecting or preventing. Recent researches in diversity field focus on Intrusion Detection Systems based on the dynamic methods, stochastic models and game theory to foresee the strategies of attacks. Elike Hodo and Xavier Bellekens drew all kind of detection methods in IDSs in a total diagram. IDS/IDPS offerings are generally categorized into two types of solutions: host-based intrusion detection systems (HIDS) and network-based intrusion detection systems (NIDS) [2]. An NIDS may incorporate one of two (or both) types of intrusion detection in their solutions: signature-based and anomaly-based. The most famous IDSs are [2-3]: 1- Snort, an open source NIDS solutions that uses both signature-based intrusion detection as well as anomaly-based methods. 2- Suricata, is a direct competitor to Snort and employs a signature-based methodology, rule/policy driven security, and anomaly-based approach for detecting intrusions. 3- Bro IDS, uses anomaly-based intrusion detection, and is usually employed in conjunction with Snort, as the two complement each other quite nicely. 4- OpenWIPS-ng, a popular open-source project for wireless security with features and functionality found. It is signature-based intrusion detection technology consists of sensors to capture wireless traffic, a server to analyze/respond to attacks, and a GUI for easy management. 5- Security Onion, is actually an Ubuntu-based Linux distribution for IDS and network security monitoring (NSM). 6- Ossec is a free and open source host based IDS that performs varied tasks like log analysis, integrity checking, Windows registry monitoring, rootkit detection, time-based alerting and active response. 7- Open Source Tripwire is a host based intrusion detection system focusing on detecting changes in file system objects. 8- AIDE (Advanced Intrusion Detection Environment) is regarded as one of the most powerful tools for monitoring changes to UNIX or Linux systems. Our aim in this proposal is covering the software diversity concept by joining various detectors and preventing solutions with a control unit to make decision and manage switching between different statuses based on the lowest possible risk. Areej Algaith and et.al in [4] used 4 IDSs (anomalous character distribution monitor based on the anomaly detection, GreenSQL to monitor the SQL traffic and detect the SQL injection, Apache Scalp-a signature based log analyzer and Snort) to cover web attacks and used web applications to assess the IDSs' performance. The result of this experiment is shown by Receiver Operating Characteristic (ROC) curve. This paper was a good advice for several detector and monitor applications usage in our scheme.

This proposal is organized in a mixed format. We first introduce the difference between software diversity and monoculture then describe what our plan for choosing diversity is. Section 4 provides our primary thought about proposed scheme and related vectors for each layer. Sections 5-7 are mainly approach to design our scheme and prove how it can be a justifiable about diversity. The rest of sections are advocated to evaluation parameters.

2. 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 [5], similar risks exist in network computer systems and getting a limited amount of diversity [6].

2. HOW TO REDUCE THE PROBLEM OF MONOCULTURE?

The more common a product is, the more it will suffer from infection by malware,

[1] break once, break everywhere

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.

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 [7]. Also for better evaluation of the network security and dependability, a prediction of the actions of the attacker and defenders is needed. Network security measurements involve the interactions of attackers and defenders, and this may affect the result of a measurement. Since the interaction process, game theory can be applied to predict the actions of the attackers and to determine the decisions of the defenders [8].

3.1 Related Works

Xiannuan Liang and Yang Xiao in [8] describe the relationship among game theory applications in network security by a diagram. Also, they imply several parameters for assessing the security and integrity such as mean time to failure (MTTF), mean time to first failure (MTFF), mean time between failures (MTBF), mean time to next failure (MTNF) and risk and the Price of Anarchy (POA) to evaluate the effectiveness of the

[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.

systems in terms of the security. The end of this paper emphasizes on the classification of models in game theory as a table. In [9-11] the authors use the diverse defender with priority between its parts. They expose their systems in a dependable graph to the relationships between normal status and vulnerable estate and ruined system which needs repairing or configuring again by administrator. Saran Neti in [7] 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:

Table 1. The aims of attacker for intrusion and related defenses

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
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:

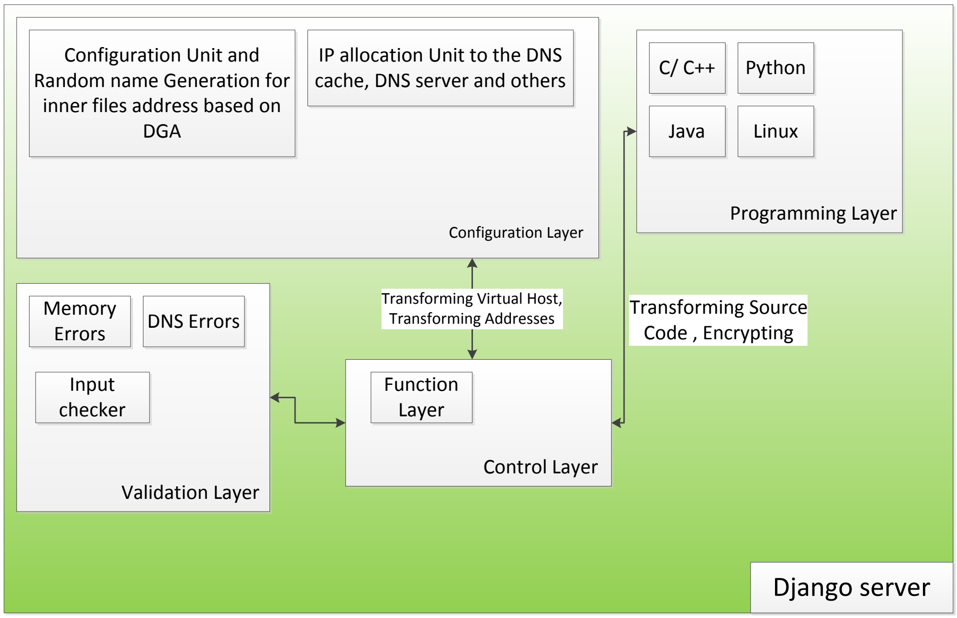
1- Using multi-tier [5]:
   Different configuration blocks besides different data centers, servers and libraries.
2- Using code transformer [12]
3- Modularity [13]:
   Separate inner parts of server (DNS caches from DNS servers).
   This solution help us save incoming DNS data from outgoing DNS data and conversely.
4- Using check threat list [14]:

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
Memory Errors
DNS Errors
Input checker
Validation Layer
Transforming Virtual Host, Transforming Addresses
Transforming Source Code , Encrypting
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

This table show us that our system consists of n Layers, m Tasks, I Inputs, j Vulnerabilities, w Global Distributions and threatened by q Attacks.
The proper equation that indicate the relevance between Layers and their Tasks:

$$Layer_p \propto \{Task_t \big|_{1 \le t \le m_p}\}; p = 1,2,3,4$$

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:

<u>Layers:</u>

- Control Layer (L1):

<u>Tasks:</u>

  - Choosing the best option and declaring the command (Switching); (t_1l_1)
  - Monitoring and Improving Functionality: *(Controlling options like EDNS0, RES_USE_EDNS0 or RES_USE_DNSSEC)*; (t_2l_1)

$$TL_1 = \{t_1l_1, t_2l_1, t_3l_1, .., t_{m_1}l_1\}$$

<u>Inputs:</u>

$$IL_1 = \{I_1l_1, I_2l_1, I_3l_1, .., I_{i_1}l_1\}$$

<u>Vulnerabilities:</u>

$$VL_1 = \{v_1l_1, v_2l_1, v_3l_1, .., v_{j_1}l_1\}$$

<u>Attacks:</u>

$$A_1 = \{a_1l_1, a_2l_1, a_3l_1, .., a_{n_1}l_1\}$$

<u>Global Distribution:</u>

$$GDL_1 = \{gd_1 l_1, gd_2 l_1, gd_3 l_1, .., gd_{w_1} l_1\}$$

- Validation Layer (L2):

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

$$TL_2 = \{t_1 l_2, t_2 l_2, t_3 l_2, .., t_{m_2} l_2\}$$

Inputs:

$$IL_2 = \{I_1 l_2, I_2 l_2, I_3 l_2, .., I_{i_2} l_2\}$$

Vulnerabilities:

$$VL_2 = \{v_1 l_2, v_2 l_2, v_3 l_2, .., v_{j_2} l_2\}$$

Attacks:

$$A_2 = \{a_1 l_2, a_2 l_2, a_3 l_2, .., a_{n_2} l_2\}$$

Global Distribution:

$$GDL_2 = \{gd_1 l_2, gd_2 l_2, gd_3 l_2, .., gd_{w_2} l_2\}$$

- Configuration Layer (L3):

  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

$$TL_3 = \{t_1 l_3, t_2 l_3, t_3 l_3, .., t_{m_3} l_3\}$$

Inputs:

$$IL_3 = \{I_1 l_3, I_2 l_3, I_3 l_3, .., I_{i_3} l_3\}$$

Vulnerabilities:

$$VL_3 = \{v_1l_3, v_2l_3, v_3l_3,..., v_{j_3}l_3\}$$

Attacks:

- Resolver Redirection Attack
- DNS Rebinding Attack

$$A_3 = \{a_1l_3, a_2l_3, a_3l_3,..., a_{n_3}l_3\};$$
$$\mathrm{Re}\,solver - \mathrm{Re}\,direction - Attack = a_1l_3$$
$$DNS - \mathrm{Re}\,binding - Attack = a_2l_3$$

Global Distribution:

$$GDL_3 = \{gd_1l_3, gd_2l_3, gd_3l_3,..., gd_{w_3}l_3\}$$

- Programming Layer (L4):

  Tasks:

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

$$TL_4 = \{t_1l_4, t_2l_4, t_3l_4,..., t_{m_4}l_4\}$$

Inputs:

$$IL_4 = \{I_1l_4, I_2l_4, I_3l_4,..., I_{i_4}l_4\}$$

Vulnerabilities:

$$VL_4 = \{v_1l_4, v_2l_4, v_3l_4,..., v_{j_4}l_4\}$$

Attacks:

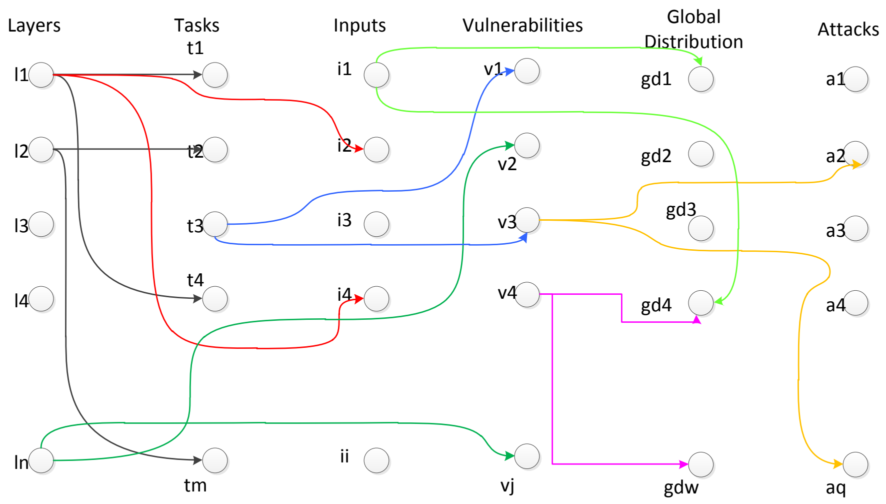
$$A_4 = \{a_1l_4, a_2l_4, a_3l_4,..., a_{n_4}l_4\}$$

Global Distribution:

$$GDL_4 = \{gd_1l_4, gd_2l_4, gd_3l_4,..., gd_{w_4}l_4\}$$

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-

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

T

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

5. WHAT IS YOUR HYPOTHESIS?

The first assumption of my proposal is that a comprehensive DNS service which consists of known parts that are used in DNS servers for implementing DNS protocol in application layer is divided to the implied Layers in previous section to detect feasible DNS attacks and replace parts with each other to reduce the impact of vulnerabilities.

The second presumption of this proposal is using dynamic switching between variants of IDSs, like [4] and different implied Layers of detectors to prevent the propagation of DNS attacks.

6. WHAT KIND OF DATA DO YOU NEED TO COLLECT?

To implement my proposal I need to gather DNS logs which include suspicious requests and clean requests.

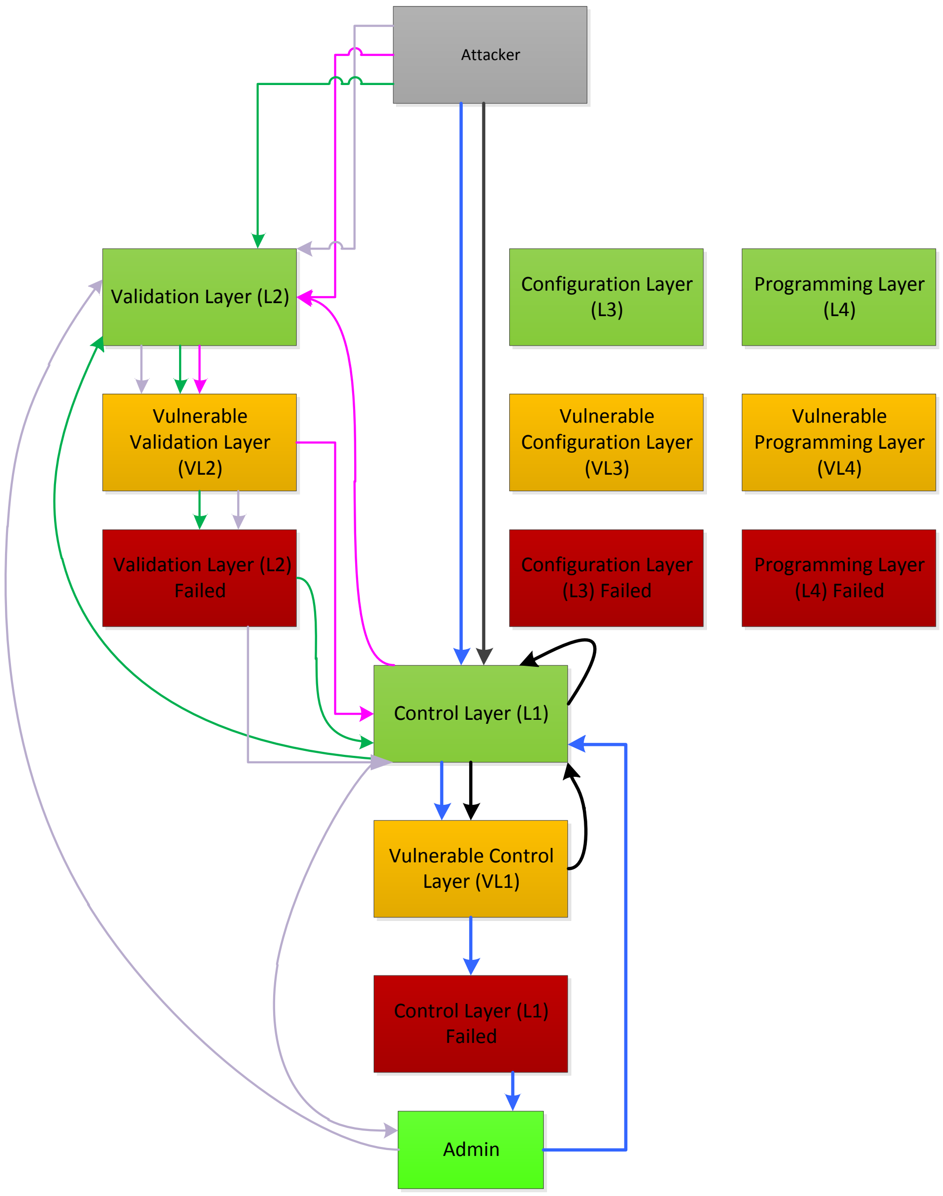
The list of my necessity about data:

1- list of all possible components for all layers (to switch among them like different IDSs in the control unit for solving faster)
2- list of the vulnerabilities of each part
3- list of feasible attacks that are related to each part
4- list of solutions to remedy the vulnerable statuses

7. HOW CAN YOU USE GAME THEORY TO VALIDATE THE IMPACT OF DIVERSIFICATION?

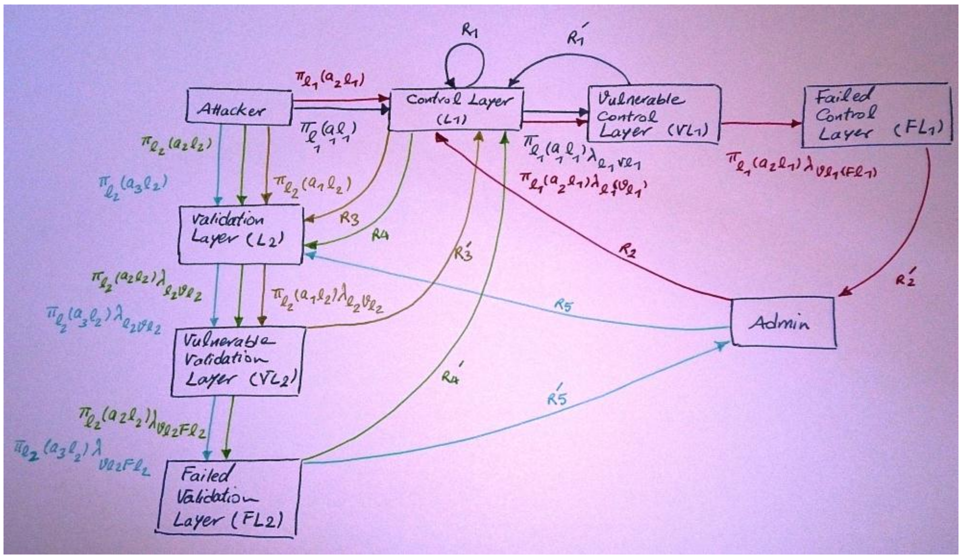
To expose my proposal, both of views can be shown. Firstly, a total system that faces attacks then predict its statuses based on vulnerabilities that may be happen to it. Secondly, use binary code to allocate each of them to each part of system then predict the related attacks and their strategies to each part. The first way can be run as a black box but the second procedure will consider to the various parts and the diversification will be shown explicitly.

Thus, the first step in my implementation is introducing all Layers and their elements with their tasks then the related vulnerabilities and possible attacks which are brought in section 4.2. the second step is drawing the graph of system and showing its statuses which may has, while the attacker's strategy focuses on it. In next figure (Attacker's and Defender's Game) I draw all Layers in three statuses, normal by green color, vulnerable with orange and failed status with dark red. If the control layer _the main layer to control all other options_ is under the direct attack then has become failed, the merely solution for returning it to the basic status will be in the hands of admin. In the below figure, all the feasible defense strategies for one of the layers are drawn. These stages will be the same for another layers, configuration layer and programming layer


Attacker
Validation Layer (L2)
Configuration Layer (L3)
Programming Layer (L4)
Vulnerable Validation Layer (VL2)
Vulnerable Configuration Layer (VL3)
Vulnerable Programming Layer (VL4)
Validation Layer (L2) Failed
Configuration Layer (L3) Failed
Programming Layer (L4) Failed
Control Layer (L1)
Vulnerable Control Layer (VL1)
Control Layer (L1) Failed
Admin

The associated formula for illustrating the probabilities in this stochastic model:
Purple line:

This line says the first process that one attack may affect the layer. This process leads the basic form of the layer to the vulnerable one. Then the control unit will find this level of attack by checking frequently. Different IDSs that are used in this unit will help us to find the reason of attack and by helping defender solutions will reduce the risk of attack. But if the level of attack is more highly or the control unit would not be punctual, the status of vulnerable layer will be altered to the failed (green line). In this way the checker of control unit will alarm the admin and accelerate its repairing process simultaneously. Because of the profound worth of each layer and the failure status which can cause the substantial destruction, admin and control unit do the parallel defense procedure (gray and green lines).The aforementioned above will be possible for two other layers.
Black line:
Black and blue lines are related to the attacks on the control unit. Black line mentions the primary attack that can be solved by self-control unit (R) and blue line is more advanced. When our control unit is failed, the system will produce alarm (R') for admin to remedy the situation.


R_1
R_1'
$\pi_{\ell_1}(a_2\ell_1)$
Attacker
Control Layer (L1)
Vulnerable Control Layer (VL1)
Failed Control Layer (FL1)
$\pi_{\ell_1}(a_1\ell_1)$
$\pi_{\ell_2}(a_2\ell_2)$
$\pi_{\ell_2}(a_3\ell_2)$
$\pi_{\ell_2}(a_1\ell_2)$
$\pi_{\ell_1}(a_1\ell_1)\lambda_{\ell_1 v\ell_1}$
$\pi_{\ell_1}(a_2\ell_1)\lambda_{\ell_f v\ell_1}$
$\pi_{\ell_1}(a_2\ell_1)\lambda_{v\ell_1(F\ell_1)}$
Validation Layer (L2)
R3
R4
R_3'
R_2
R_2'
$\pi_{\ell_2}(a_2\ell_2)\lambda_{\ell_2 v\ell_2}$
$\pi_{\ell_2}(a_1\ell_2)\lambda_{\ell_2 v\ell_2}$
$\pi_{\ell_2}(a_3\ell_2)\lambda_{\ell_2 v\ell_2}$
R5
Admin
Vulnerable Validation Layer (VL2)
R_4'
$R5'$
$\pi_{\ell_2}(a_2\ell_2)\lambda_{v\ell_2 F\ell_2}$
$\pi_{\ell_2}(a_3\ell_2)\lambda_{v\ell_2 F\ell_2}$
Failed Validation Layer (FL2)

Vulnerabilities vector and attacks vector:

$$VL_1 = \{v_1l_1, v_2l_1, v_3l_1, .., v_{j_1}l_1\}$$

$$a_1l_1 \in A_1; A_1 = \{a_1l_1, a_2l_1, ..a_{n_1}l_1\}$$

$$a_1l_2 \in A_2; A_2 = \{a_1l_2, a_2l_2, ..a_{n_2}l_2\}$$

$$a_1l_3 \in A_3; A_3 = \{a_1l_3, a_2l_3, ..a_{n_3}l_3\}$$

$$a_1l_4 \in A_4; A_4 = \{a_1l_4, a_2l_4, ..a_{n_4}l_4\}$$

Reward and cost values:
r_{n1}={(a_1l_1 | undetected), (a_2l_1 | undetected), (a_3l_1 | undetected),.., ($a_{n1}l_1$ | undetected)}

r_{n1}={(a_1l_1 | detected), (a_2l_1 | detected), (a_3l_1 | detected),.., ($a_{n1}l_1$ | detected)}

r_{n2}={(a_1l_2 | undetected), (a_2l_2 | undetected), (a_3l_2 | undetected),.., ($a_{n2}l_2$ | undetected)}
r_{n2}={(a_1l_2 | detected), (a_2l_2 | detected), (a_3l_2 | detected),.., ($a_{n2}l_2$ | detected)}

r_{n13}={(a_1l_3 | undetected), (a_2l_3 | undetected), (a_3l_3 | undetected),.., ($a_{n3}l_3$ | undetected)}
r_{n3}={(a_1l_3 | detected), (a_2l_3 | detected), (a_3l_3 | detected),.., ($a_{n3}l_3$ | detected)}

r_{n4}={(a_1l_4 | undetected), (a_2l_4 | undetected), (a_3l_4 | undetected),.., ($a_{n4}l_4$ | undetected)}
r_{n4}={(a_1l_4 | detected), (a_2l_4 | detected), (a_3l_4 | detected),.., ($a_{n4}l_4$ | detected)}

Transition probabilities between the game states:

$$\rho_{l_1vl_1} = \frac{\lambda_{l_1vl_1}}{\lambda_{l_1vl_1} + R_1'}; v = v_1, v_2, .., v_{j_1}$$

$$\rho_{vl_1fl_1} = \frac{\lambda_{vl_1fl_1}}{\lambda_{l_1vl_1} + \lambda_{vl_1fl_1} + R_1' + R_2'}$$

$$\rho_{l_2vl_2} = \frac{\lambda_{l_2vl_2}}{\lambda_{l_2vl_2} + R_3'}; v = v_1, v_2, .., v_{j_2}$$

$$\rho_{vl_2fl_2} = \frac{\lambda_{vl_2fl_2}}{\lambda_{l_2vl_2} + \lambda_{vl_2fl_2} + R_3' + R_4' + R_5'}$$

$$\rho_{l_3vl_3} = \frac{\lambda_{l_3vl_3}}{\lambda_{l_3vl_3} + R_6'}; v = v_1, v_2, .., v_{j_3}$$

$$\rho_{vl_3fl_3} = \frac{\lambda_{vl_3fl_3}}{\lambda_{l_3vl_3} + \lambda_{vl_3fl_3} + R_6' + R_7' + R_8'}$$

$$\rho_{l_4vl_4} = \frac{\lambda_{l_4vl_4}}{\lambda_{l_4vl_4} + R_9'}; v = v_1, v_2, .., v_{j_4}$$

$$\rho_{vl_4fl_4} = \frac{\lambda_{vl_4fl_4}}{\lambda_{l_4vl_4} + \lambda_{vl_4fl_4} + R_9' + R_{10}' + R_{11}'}$$

Solving the game model:

$$\Gamma = \begin{bmatrix} r_{n_1}(A_1 \mid undetected) + \sum_{k=1}^{j_1} (\rho_{l_1 v_k l_1} + \rho_{v_k l_1 fl_1}) \pi_{l_1}(a_k l_1) & r_{n_1}(A_1 \mid detected) \\ r_{n_2}(A_2 \mid undetected) + \sum_{k=1}^{j_2} (\rho_{l_2 v_k l_2} + \rho_{v_k l_2 fl_2}) \pi_{l_2}(a_k l_2) & r_{n_2}(A_2 \mid detected) \\ r_{n_3}(A_3 \mid undetected) + \sum_{k=1}^{j_3} (\rho_{l_3 v_k l_3} + \rho_{v_k l_3 fl_3}) \pi_{l_3}(a_k l_3) & r_{n_3}(A_3 \mid detected) \\ r_{n_4}(A_4 \mid undetected) + \sum_{k=1}^{j_4} (\rho_{l_4 v_k l_4} + \rho_{v_k l_4 fl_4}) \pi_{l_4}(a_k l_4) & r_{n_4}(A_4 \mid detected) \end{bmatrix}$$

Solving this matrix means to calculate the strategies for the attacker and defender (in my proposed system).

8. WHAT KIND OF PARAMETERS DO YOU NEED TO TUNE?

The most important issue in game theory is tuning the defender's strategies (R) to respond faster to the R' requests. The other options of game theory that should be tuned are rewards and cost, r_{n_i} .

(The details of these options will be explained soon)

9. HOW WOULD YOU ASSESS THE QUALITY OF YOUR SOLUTION?

For evaluating my scheme I will inaugurate the operational lab with two computers which one of them has an attacker role and the other is used as a DNS resolver with four-layer. Before starting attacks we assume that she is a real user who wants to check her system without any suspicious manner and request her domain to the DNS Server. Therefore we will have the pure traffic which ensured us that it is white. After gathering white data and filling the relation table in DNS server our lab is ready to run a known DNS attack (which mentioned in previous sections). In this stage I can see how my system will work against the black traffic. To evaluate this labor, first, I can achieve the time consumption for producing the first alarm. Then I can record the strategy of my system for choosing the best solution based on the stochastic parameters and compare it with my previous calculation. The last method that comes to my mind is drawing the ROC curve and comparing the ability of my detector to other existent methods.

10. WHAT ARE METRICS TO ASSESS THAT YOUR HYPOTHESIS HOLD?

To assess my scheme, there are four metrics in [15] such as the mean time to first failure (MTFF), the mean time to failure (MTTF), the mean time between failures (MTBF) and the mean time spent in the good states (MUT) that can be used for evaluating my scheme or using the ROC curve to compare with the previous works.

11. HOW DO YOU BELIEVE THAT YOUR SOLUTION CAN IMPROVE SECURITY?

The best answer to this question is in the drawn diagram in section 7. In this image we can see that each attempt of attacker should be down on at least two parts. Thus our diversity scheme increases the reliability of DNS resolver and forces attacker to use more complex strategies for intrusion.

(This section needs more research and investigation about my scheme)

12. HOW IS YOUR SOLUTION GOOD?

One of the hallmarks of the proposed method is that it comprises various units with different duties associated with their roles in DNS service. Each unit can has independent IP or specific glue, the benefit of this way is puzzling attacker and preventing system from her intrusion. Another advantage of this method is switching between different parts easily by control unit’s help. The last privilege of this method is to calculate the probability of an attack occurrence and strengthen controls of associated part.

13. WHAT KIND OF EXPERIMENT CAN YOU DESIGN TO VALIDATE THE BENEFITS OF DIVERSITY?

One of the approaches to find the privileges of diversity is comparing the proposed scheme by the ones that are simpler and have less option or compare with the states of my scheme while I assume one layer like a simple DNS resolver which is used by novices.

14. TRADE-OFF BETWEEN 3 OPTIONS?

It is necessity to find a trade-off between:

- The amount of diversity
- The cost of production
- Maintenance and the overall security

This is why increasing randomness in the execution of systems is often used as a form of dynamic diversification.

(This section will be filled soon)

15. 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. Elike Hodo and et.al, “Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey”, 2017.
2. *https://www.upguard.com/articles/top-free-network-based-intrusion-detection-systems-ids-for-the-enterprise.*
3. *http://opensourceforu.com/2017/04/best-open-source-network-intrusion-detection-tools/*
4. Areej Algaith and et.al, “Diversity with Intrusion Detection Systems: An Empirical Study”, 16th IEEE International Symposium on Network Computing and Application, Nov 2017.
5. Simon Allier and et.al, “Multi-tier diversification in web-based software application”, *IEEE Software*, Vol.32, 2015, pp.83-90.
6. Bruce schneier, “the dangers of a software monoculture”, Meeting cloud computing compliance mandates, 2010.
7. 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.
8. Xiannuan Liang and Yang Xiao, “Game Theory for Network Security”, IEEE Communication Survey & Tutorials, Vol.15, No.1, first quarter 2013.
9. Zhiyong Luo and et.al, “Analysis and Optimization of System Intrusion Tolerance Capacity Based on Markov”, International Journal of Network Security, Vol.19, No.6, 2017.
10. Karin Salhammar and et.al, “Towards a Stochastic Model for Integrated Security and Dependability Evaluation”, International Conference on Availability, Reliability and Security IEEE, 2006.
11. Ibidunmoye EO, “Game-theoretic Scenario for Modeling the Attacker-Defender Interaction”, Journal of Computer Engineering & Information Technology, 2013.

*12.*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.*

13. https://cr.yp.to/djbdns/separation.html

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

15. John A. Buzacott, “Markov approach to finding failure times of repairable system”, IEEE Transactions on Reliability, R-19:128-134, November 1970.