NIDS : performance (Snort. Algorithmique)

1 Introduction:

a) Les IDS:

Un système de détection d'intrusion (IDS) est un mécanisme visant à détecter d'éventuelles intrusions dans le système/réseau. Les menaces sont alors signalées à l'administrateur (monitoring, log). Certains IDS (les IPS) sont capables d'effectuer une action de contre-mesure afin de tenter de contrer la menace (drop de paquets, blocage de port ...).

Il existe 3 types d'IDS:

- Host-based intrusion detection systems (HIDS):

  Se place sur la machine hôte, il va scanner les activitées du système. Il repose en grande partie sur les fichiers de log qui lui permetrons d'en déduire des menaces par corélation d'évenements.

- Network-based intrusion detection systems (NIDS):

  Se place générallement après le firewall (afin de recevoir un flux deja préfiltré par le firewall et ne pas être débordé). Il reçoit tout le trafic réseau puis l'analyse afin de détécter d'éventuelles menaces.

  Les NIDS vont inspecter chaque paquet et vérifier si ils comportent une séquence coincidant avec une signature du set configuré.

  Ils fonctionnent en 3 étapes:

  - mise du paquet dans un buffer
  - signature matching (par pattern-matching)
  - signaler l'intrusion (et agir dans le cas des NIPS)

- Hybrid intrusion detection systems:

  C'est un HIDS qui va aussi surveiller les activitées réseau.

b) SNORT:

Snort est un NIDS et NIPS open source créé en 1998 par Martin Roesch.

Il est l'un des NIDS open source les plus répendu grâce à sa grande communauté.

Il permet de faire une analyse en temps réel du trafic.

Il doit donc effectuer l'analyse sur les paquets à une vitesse proche de la vitesse de transmission afin de ne pas trop affecter le débit (online analysis).

Il possède un système de règles alimentée par la communité. Ces règles comportent les informations du protocole à tester, les sinatures ainsi que l'action à executer en cas de "match".

La structure des protocoles est elle aussi définis via un système de règle.

Il y a environ 3500 rules dans la version community actuelle (2.9.7).

c) Problématiques:

L'opération la plus couteuse pour un NIDS est l'étape de "signature matching".

Elle se fait en 2 étapes:

- Classification des paquets: En examinant les header fields, on identifie quel sont les règles à tester sur ce paquet.
- Deep packet inspection: On vérifie si le contenu du packet correspond aux différentes signatures des règles configurées. Pour se faire, deux techniques sont employées:
  - String matching
  - Regular expression matching

Dans le cas de Snort qui fait de l'analyse en temps réel, on ne peut pas se permettre d'avoir des temps de calcul trops long pour l'analyse des paquets. Il faut arriver à tous les traités à une vitesse proche de celle du débit afin de ne pas trops le faire chuter.

D'où la nécessité de trouver un algorithme de patern matching le plus performant possible.

2 Les algorithmes de pattern-matching:

A ces débuts, Snort faisait le patern matching par brute force, ce qui le rendait très lent.

Il a ensuite rapidement implémenté l'algorithme de Boyer-Moore. Cet algorithme est extrèmement efficace lorsque l'on recherche une seule chaine de caractère.

Cpendant il est très courrant que de nombreuses règles doivent être testées sur un même paquet.

Il va ensuite utiliser l'algorithme de Aho-Corassick qui est bien plus performant pour tester un grand nombre de signature.

a) Brute Force:

Cela consiste à aligner le 1er caractère de la chaîne à tester avec le 1er de la signature, puis de tester tous les carctères de la signature jusqu'à trouver un mismatch. Ensuite on réitère en décalant de 1 la signature jusqu'à avoir un match ou avoir parcouru toute la chaîne.

(cf illustration 1)

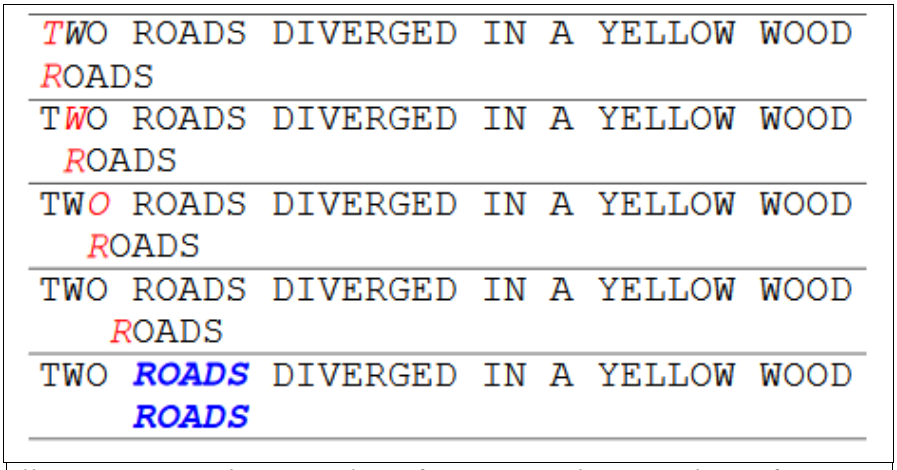


Illustration 1: Schéma explicatif string matching par brute force

src: https://www.cs.purdue.edu/homes/ayg/CS251/slides/chap11.pdf

b) Boyer Moore:

Algorithme très performant pour la recherche d'une unique chaîne.

Il consiste à optimiser les comparaisons de caractères en profitant de ce que l'on a deja comparé.

Pour cela il effectue des sauts lors de mismatch ou de presence de suffixe en sous-chaîne.

Il nécessite un calcul des tables des sauts au préalable.

Il procède généralement par test des caractères de droite à gauche.

Cas du mismatch:

Lorsque l'on tombe sur un mismatch d'un seul caractère, on peut directement sauter à la prochaine occurence de ce caractère. Pour cela, on peut calculer une table de saut indiquant pour chaque caractère de l'alphabet le nombre de décalage à effectuer avant de trouver sa prochaine occurence.

Pour la chaîne "WIKIPEDIA" on obtient:

Caractère de l'alphabet	Distance à la fin de clé
I	1
D	2
E	3
P	4
K	6
W	8
autres caractères	9

Illustration 2: Table de saut

Cas du suffixe sous-chaîne:

Prenons par exemple, la chaîne à tester "ANPANMAN" et la chaîne clef "PAN".

Lorsque l'on arrive à un match des suffixes "AN", on arrive à un mismatch de "PAN" et "MAN". Si l'on agissait naïvement, il nous faudrait décaler 3 fois en retestant tous les caractères avant d'atteindre le match complet.

On remarque que l'on pouvait prévoir au vu de la constitution de "ANPANMAN" qu'il nous fallait faire au moins 3 décalages. En effet le suffixe "AN" est une sous-chaîne de "ANPANMAN", on peut donc lors d'un match de "AN" sauter directement à la précedente occurence de cette sous-chaîne.

En faisant ainsi pour tous les préfixes de "ANPANMAN" on obtient la table suivante:

Indice							Motif suivant								Décalage obtenu
							A	N	P	A	N	M	A	N	
0													~~N~~		1
1					~~A~~	N									8
2									~~M~~	A	N				3
3					~~N~~	M	A	N							6
4				~~A~~	N	M	A	N							6
5			~~P~~	A	N	M	A	N							6
6		~~N~~	P	A	N	M	A	N							6
7	~~A~~	N	P	A	N	M	A	N							6

Illustration 3: Seconde table de saut

src: https://fr.wikipedia.org/wiki/Algorithme_de_Boyer-Moore

c) Aho-Corasick:

Très efficace pour un set de signature.

Il consiste à construire un automate à état fini permettant de tester une liste de signature en parcourant une seule fois la chaîne.

Afin de le construire, on va dresser un arbre à partir de la liste des signatures (voir illustration 4).

Ensuite, on va relier chaque etat avec tous les états de ces suffixes. Par exemple, "he" correspond à l'état 2 et est un suffixe de "she" (état 5). On relie donc 5 vers 2.

Cela permet de profiter habilement de ce que l'on a deja testé.

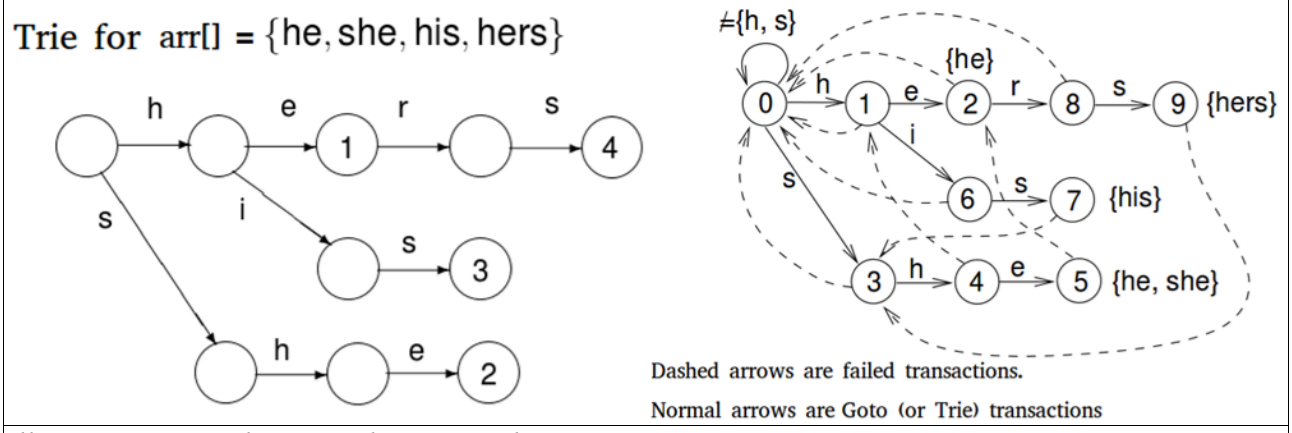


Illustration 4: Explication Aho-Corasick

3 Les pistes d'amélioration:

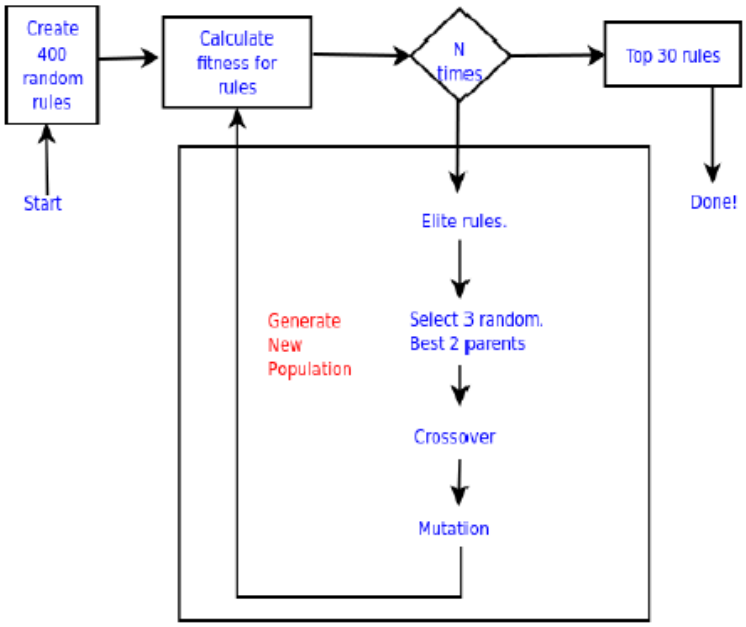
De nombreuses recherches sont en cours.

- Il y a notament Silicon Defense qui a effectué des recherches sur un nouvel algorithme combinant les avantages de celui de Boyer-Moore et de Aho-Corassick. C'est à dire appliquer l'utilisations des tables de saut au sein de l'automate de Aho-Corassick. (cf ref)
- D'autres travaux tente d'implémenter les algorithmes génétiques.

  Une méthode consiste à optimiser le set de règles selon le principe des algorithmes génétiques.

  Pour cela, on génère aléatoirement un set de règle que l'on va tester puis évaluer afin de trouver des gènes "d'élite" que l'on va utiliser pour regénérer une nouvelle population. En réitérant ainsi un grand nombre de fois on arrive à avoir un set de règle très petit et efficace.

  (pour plus de détails voir ref)



- D'autres implémentations ou améliorations d'algorithmes sont aussi envidagées, tel que l'algorithme de Wu Manber ou des optimisations d'Aho Corasick.

  (voir ref)

4 Conclusion:

Afin de contrer plus efficacement les menaces du réseau, les NIPS on été conçus. Ils se placent en coupure sur le réseau afin de pouvoir agir en réponse et ainsi tenter de contrer la menace. Cependant cette technique nécessite de traiter le flux de donné en temps réel à une vitesse la plus proche possible de la vitesse de transmission.

Cette problématique touche par exemple Snort qui à beaucoup évolué en ce sens depuis sa création.

Actuellement, les pistes d'améliorations sont loin d'être épuisées, notament avec le développement des algorithmes génétiques et de l'intélligence artificielle.

5 Références:

a) Snort:

Website: https://www.snort.org/

b) Patternes Matching:

https://fr.wikipedia.org/wiki/Algorithme_de_Boyer-Moore

http://www.geeksforgeeks.org/aho-corasick-algorithm-pattern-searching/

c) Application du pattern matching aux IDS:

- **James Kelly, "An Examination of Pattern Matching"** thèse de master en informatique,sous la direction de Dr. Paul Van Oorschot, Carleton University, Canada, 228 p.

  (très complet sur les algorithmes de pattern matching, leurs applications aux IDS et à Snort)

  Ref: https://ccsl.carleton.ca/people/theses/Kelly_Master_Thesis_06.pdf

- **Mike Fisk, "Applying Fast String Matching to Intrusion Detection"**, University of California San Diego

  (analyse des performances et présentation d'un nouvel algorithme)

  Ref: http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-01-5459

d) Performances:

- **Adeeb Alhomoud, Rashid Munir, Jules Pagna Disso, Irfan Awan, A. Al-Dhelaan, "Performance Evaluation Study of Intrusion Detection Systems"**, The 2nd International Conference on Ambient Systems, Networks and Technologies, 2011.

  (Etude comparative avec son rival: Suricata)

  http://www.sciencedirect.com/science/article/pii/S1877050911003498

e) Application des algorithmes génétiques:

- **G.V. Nadiammai and M. Hemalatha , "Handling Intrusion Detection System using Snort Based Statistical Algorithm and Semi-supervised Approach"**, Department of Computer Science, Karpagam University, India, 2013.
  https://www.semanticscholar.org/paper/Handling-Intrusion-Detection-System-using-Snort-Ba-Author-Hemalatha/60230387673e861e4cc3172ad497732d1899c48c
- Mit H. Dave, Dr. Samidha Dwivedi Sharma, "Improved Algorithm for Intrusion Detection Using Genetic Algorithm and SNORT", NRI Institute of Science & Technology, Bhopal, India, 2014.
  http://www.ijetae.com/files/Volume4Issue8/IJETAE_0814_43.pdf

f) Autres Algorithmes ou améliorations:

- **Vasudha Bhardwaj et Vikram Garg, "A Comparative Study of Wu Manber String Matching Algorithm and its Variations"**, LNCTE, Bhopal , International Journal of Computer Applications (0975 – 8887) Volume 132 – No.17, December 2015.
  http://www.ijcaonline.org/research/volume132/number17/bhardwaj-2015-ijca-907708.pdf
- **Marc Norton, "Optimizing Pattern Matching for Intrusion Detection"**, Snort IDS Team Lead at Sourcefire.
  http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.103.4663&rep=rep1&type=pdf