Detection and Threat Prioritization of Pivoting Attacks in Large Networks

(Supplementary Material)

Giovanni Apruzzese, Fabio Pierazzi, Michele Colajanni, Mirco Marchetti

Appendix A
Computational complexity of pivoting detection algorithms

We evaluate the computational complexity of the proposed pivoting detection algorithm and compare it against two alternatives: subgraph isomorphism and brute force enumeration algorithms.

Subgraph isomorphism

If we consider a pivoting path (or sequence) as a subgraph, then the pivoting detection problem could be seen as that of finding occurrences of specified subgraphs within a graph. This approach, which is known as the subgraph isomorphism problem, is *NP-complete* [1] even for static graphs without temporal edges. Hence, it is not a viable solution.

Brute force enumeration

A possible approach to pivoting detection is to enumerate all possible sequences that can be derived from existing flows, and then evaluate whether these flow sequences are consistent with a maximum propagation delay ε_{max}. The complexity of this brute force enumeration algorithm is:

$$\sum_{L=1}^{m}\left[\binom{m}{L}\cdot L!\right]\sim\Omega(2^m) \tag{1}$$

where $\binom{m}{L}$ represents the number of possible combinations (subsets) of length L given m flows; $L!$ denotes all possible permutations of such elements, and counts the number of possible re-orderings of a length L path. The computational complexity is more than exponential in the number of edges m, and is always higher than $\Omega(2^m)$. If we simplify the problem by considering only contiguous subsequences as in [2], then the complexity diminishes (that is, L^2 instead of $L!$ as a multiplicative factor in Eq. 1), but still remains more than exponential in the number of edges m.

Pivoting detection

The algorithm for pivoting detection proposed in this paper has an overall worst-case time complexity of:

$$\mathcal{O}(m^{L_{max}}\cdot\log_2(m)\cdot\tau) \tag{2}$$

where m is the number of network flows within the window W, L_{max} is the maximum pivoting tunnel length we are looking for, and τ is the maximum number of flows between any $[t, t+\varepsilon_{max}]$ interval. For small values of ε_{max} representing the common case, the parameter $\tau \ll m$, hence the complexity may be simplified as follows:

$$\mathcal{O}(m^{L_{max}}\cdot\log_2(m)) \tag{3}$$

For the demonstration, we assume that the m flows arrive in order of timestamp. The initialization phase requires $\mathcal{O}(m)$ operations to initialize the list of flow sequences with length $L=1$. The computational complexity of the initialization phase is then:

$$\mathcal{O}(m) \tag{4}$$

The number of iterations of the core part starting on line 9 of the Algorithm 1 depends on the total number of possible flow sequences with length between 1 and L_{max}. We recall that the flow sequences are included in the *PivotingSequences* list in Algorithm 1 while they are being found.

Let k_i be the number of i-length pivoting flow sequences that can be seen as the number of permutations without repetition of m flows in ordered groups of i elements [3]:

$$k_i = \frac{m!}{(m-i)!} \tag{5}$$

$$= m\cdot(m-1)\cdot\ldots\cdot(m-i+1) \tag{6}$$

$$= \mathcal{O}(m^i) \tag{7}$$

• *The authors are with the Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Italy. Email: {giovanni.apruzzese, fabio.pierazzi, michele.colajanni, mirco.marchetti}@unimore.it*

As we have to consider the total number of flow sequences of length $i = \{1, 2, ..., L_{max}\}$, we analyze $\sum_{i=1}^{L_{max}}(m^i)$ sequences, which can be approximated to the following known *geometric series* [4]:

$$\sum_{i=0}^{L_{max}} m^i = \frac{1 - m^{L_{max}+1}}{1-m} = \mathcal{O}(m^{L_{max}}) \tag{8}$$

Then, we have to consider that for each iteration on line 9, the function *ExtendPivotingPath* is executed.

In the *ExtendPivotingPath* function, we have a binary search in the sorted list of m flows, that takes $\mathcal{O}(\log_2(m))$ time. Then, if τ is the maximum number of flows between any t and $t+\varepsilon_{max}$ timestamps, we have an overall time complexity of the function *ExtendPivotingPath* equal to:

$$\mathcal{O}(\log_2(m) \cdot \tau) \tag{9}$$

From Eq. 1, Eq. 5 and Eq. 6 we obtain a worst-case complexity of:

$$\mathcal{O}(m + m^{L_{max}} \cdot \log_2(m) \cdot \tau) \tag{10}$$

where $L_{max} \ll m$ and $\tau \ll m$ (for small values of ε_{max}).

Appendix B
Dataset

We release a subset of the traffic dataset used in our paper. It consists of about 75M network flows among about 1K hosts, corresponding to two weeks of activities of a large organization. The dataset contains benign pivoting paths with no pivoting-related attacks. Each network flow reports the information presented in Section 4. For privacy reasons, we have anonymized source and destination IP addresses; to facilitate analysis, we have associated a label with each flow to denote whether it belongs to a pivoting path. Access to the dataset can be requested at the following link: *https://weblab.ing.unimore.it/pivoting/dataset*.

References

[1] A. Gupta and N. Nishimura, "Characterizing the complexity of subgraph isomorphism for graphs of bounded path-width," in *STACS*. Springer, 1996.

[2] A. Paranjape, A. R. Benson, and J. Leskovec, "Motifs in Temporal Networks," in *ACM WSDM*, 2017.

[3] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[4] T. T. Soong, *Fundamentals of probability and statistics for engineers*. John Wiley & Sons, 2004.