Probabilistic Dye-Pumping 알고리즘을 이용한 P2P 봇넷 멤버 탐지

최승환*, 박효성*, 김기창*

*인하대학교

Detecting Members of P2P Botnets Using Probabilistic Dye-Pumping Algorithm

Seung-hwan Choi*, Hyo-seong Park*, Ki-chang Kim*

*Inha University

E-mail : sh.choi.925@gmail.com, mongsiry013@hanmail.net, kchang@inha.ac.kr

요 약

봇넷은 악성 코드에 의해 감염된 봇 호스트들로 이루어진 네트워크를 의미한다. 보편적으로 쓰이고 있는 Centralized 봇넷의 경우 상대적으로 C&C 서버의 위치 탐지가 용이한 반면, P2P 봇넷은 여러가지 회피 기술로 인해 봇넷의 구조를 파악하기 어렵다. 본 논문에서는 라우터를 기준으로 내부, 외부 네트워크를 구분하고 내부와 외부 네트워크의 송수신 패킷의 경로, 감염 확률을 통해 봇넷을 탐지하는 방법에 대해 연구하였다. 본 연구에서는 기존의 P2P 봇넷 탐지 방법인 Dye-Pumping의 한계를 개선하였으며, 이는 단위 네트워크 내의 P2P 봇 호스트들을 탐지하고 이들의 활동을 사전에 방지하여 P2P 봇넷이 외부로 확산되는 것을 막을 수 있는 기술 마련의 기초로써 사용될 수 있을 것으로 기대된다.

ABSTRACT

Botnet is a network that consists of bot hosts infected by malware. The C&C server of centralized botnet, which is being used widely, is relatively easy to detect, while detecting P2P botnet is not a trivial problem because of the existence of many avoiding techniques. In this paper, we separate the network into inner and outer sub-network at the location of the router, and analyze the method of detecting botnet using path of packet and infection probability. We have extended Dye-Pumping algorithm in order to detect P2P botnet members more accurately, and we expect that the analysis of the results can be used as a basis of techniques that detect and block P2P botnet in the networks.

키워드

P2P Botnet, Dye-Pumping, Probabilistic, Network Security

Ⅰ. 서 론

최근 악성 봇넷을 통해 조직이나 개인의 보안을 위협하는 공격이 급증하고 있다. 봇넷(Botnet)이란 악성코드에 감염 된 봇들이 이루고 있는 네트워크 망을 의미하고, 봇(Bot)은 봇 마스터가 전달한 커맨드를 실행하는 호스트들로써, 좀비라고도 일컬어진다. 봇넷은 봇 마스터가 트로이 목마, 분산 서비스 거부(DDoS : Distributed Denial of Service) 공격을 수행하는데 이용되며, 인터넷 상에서 가장 많이 사용되는 공격 플랫폼으로 발전하여 왔다.

봇넷은 크게 Centralized 방식과 P2P 방식으로 구분할 수 있다. Centralized 방식의 봇넷은 중앙 C&C 서버가 존재한다[1]. 봇 마스터가 C&C 서버로 커맨드를 전송하면 봇넷의 봇 호스트들은 C&C 서버로부터 커맨드를 전달 받고 해당 커맨드를 수행한다. 이런 Centralized 방식의 봇넷의 경우 네트워크 관리자가 C&C 서버의 위치를 탐지 한 후 내부 네트워크와 C&C 서버와의 연결을

차단함으로써 봇의 활동을 막을 수 있다.

반면, P2P 봇넷은 중앙 C&C 서버가 따로 존재하지 않는다. 봇 호스트들은 P2P 방식으로 네트워크 망을 구성하고 있으며, 각 호스트는 자신 이외의 일부 봇넷 구성원의 IP 리스트를 가지고 있다. 어느 한 봇 호스트가 봇 마스터로부터 커맨드를 입력 받는다면 이 호스트는 입력 받은 커맨드를 자신의 리스트에 있는 다른 호스트들에게도 전달한다. 다른 봇으로부터 커맨드를 전달 받은 봇 호스트 또한 자신의 IP 리스트에 있는 봇들에게 커맨드를 전달하여 봇넷의 멤버들은 C&C 서버가 없이도 커맨드를 전달 받게 된다. 따라서, 네트워크 관리자가 봇넷의 구조를 파악하기 어렵고, 봇넷의 활동을 저지시키는 것도 쉽지 않다.

본 논문에서는 2장에서 살펴 볼 Dye-Pumping 알고리즘[2]과 이를 통해 생성된 Edge가 감염되었을 확률을 이용하여 감염된 호스트와 감염되지 않은 호스트들을 구분하였다. 기존의 Dye-Pumping 알고리즘은 각 호스트들이 가지는 Mutual Contact를 통해 해당 호스트의 감염 가능성 여부를 결정하는데 각 Mutual Contact들은 주로 봇 호스트들 사이에 생성된다는 전제를 하고 있다. 그러나 실제 환경에서는 봇 호스트들 뿐만 아니라 일반 호스트들 사이에서도 많은 Mutual Contact가 존재 하므로 경우에 따라 알고리즘 결과의 신뢰도가 낮은 경우가 발생한다. 본 연구에서는 각각의 Mutual Contact에 확률을 부여하여 이러한 Dye-Pumping 알고리즘의 문제점을 개선하였고, 새로운 알고리즘에 Probabilistic Dye-Pumping 이라는 이름을 부여하였다.

본 논문의 실험 결과, 각 알고리즘을 100번 수행하였을 때 Dye-Pumping 알고리즘은 약 70%의 검출도와 88%의 정확도를 보였고, Probabilistic Dye-Pumping 알고리즘은 약 80%의 검출도와 96%의 정확도를 보였다.

본 논문의 2장에서는 기존의 P2P 봇넷 탐지 방법과 그 한계를 살펴 볼 것이고, 3장에서는 Probabilistic Dye-Pumping 알고리즘을 소개한다. 4장에서는 기존의 Dye-Pumping 알고리즘과 Probabilistic Dye-Pumping 알고리즘의 수치적인 결과를 비교하고, 5장에서는 결론과 향후 전망에 대하여 언급한다.

II. 관련연구

II-I Network stream

P2P 노드 탐지 알고리즘으로 각 노드의 network stream을 분석하여 한 무리의 P2P 노드들을 찾아 해당 노드의 감염 여부를 확인하는 방식이다. P2P 프로그램은 그들만의 P2P 프로토콜을 가지며, 감염된 봇 사이에서는 데이터 교환이 자주 일어나기 때문에 이에 대한 통계를 내고, 이를 통해 각각의 봇 사이에서 수행되는 악성 프로그램의 정보 수집이 가능하다. 통계 결과 의심이 가는 network stream을 통해 해당 프로그램이 봇넷에 의한 것인지를 판단한다[3].

II-II Contact tracing

각 노드들의 contact 중, 일정 비율 이상으로 연결되는 contact를 추적하고, 해당 contact와 연관성이 있으며 의심이 가는 특성을 보이는 다른 contact들도 추적하여 contact tracing chain을 형성한다. 이렇게 형성된 chain의 길이가 미리 정해놓은threshold에 이르면 해당 chain에 속한 노드들은 모두 감염된 봇으로 간주한다[4].

II-III Dye-pumping 알고리즘

Dye-pumping 알고리즘[2]에서는 내부 네트워크의 어느 한 봇이 사전에 알려져 있을 때 적용 가능한 알고리즘이다. 내부 네트워크에 호스트 A, B가 있다고 하고 외부 네트워크에 호스트 C가 있다고 할 때, 만약 호스트 A와 C가 통신을 하고 B와 C가 통신을 하면 그림 2.1과 같이 호스트 C는 호스트 A와 B의 mutual contact가 된다. 만약 호스트 A가 감염된 봇이라면 간접적인 경로, 즉 mutual contact를 통해 B에게 데이터를 전송하였을 가능성이 있다.

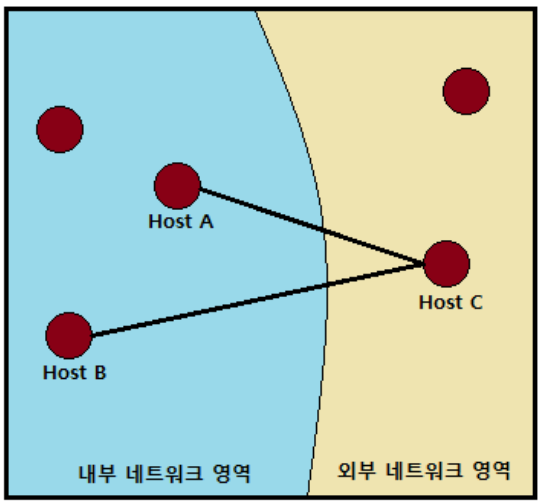

그림 2.1 Mutual contact

호스트 A와 B의 mutual contact는 1개 이상일 수 있다. 이 mutual contact 개수의 의미는 중요하다. 만약 호스트 A가 호스트 B와의 mutual contact 개수가 호스트 B가 아닌 내부 네트워크의 다른 호스트와의 mutual contact 개수보다 많으면 호스트 B는 호스트 A와 P2P로 연결되어 있을 가능성이 크다. 이 아이디어를 기반으로

내부 네트워크에 P2P로 연결되어 있는 봇넷의 멤버들을 탐지하는 방법이 Dye-pumping 알고리즘이다.

Dye-pumping 알고리즘 수행을 위해 필요한 계수가 있다.

privacy-threshold : Google, Yahoo 등 유명한 사이트의 경우 내부 네트워크들과 많은 통신을 한다. 이런 사이트들은 봇넷의 멤버일 가능성이 낮으므로 mutual contact에 포함되지 않도록 해야한다. 외부 네트워크의 호스트가 몇 개의 내부 네트워크 호스트와 통신하였는지 확인하고, 이 개수가 privacy-threshold를 초과하면 해당 외부 네트워크 호스트는 mutual contact에 포함시키지 않는다.

행렬 **T** : **T(i,j)**는 내부 네트워크의 노드 i와 j가 갖고 있는 Mutual contact의 개수와 다른 계수들에 의해서 정해지지만 본 논문에서는 mutual contact의 개수만으로 행렬 T를 나타낼 것이다.

벡터 **L** : L(i) = 1 (if s = 1)
L(i) = 0 (elsewhere)
내부 네트워크 i가 만약 seed node라면 1이고 그렇지 않으면 0을 갖는다.

```
Dye_Pumping(E, s, maxIter)
T ← computeTransitionMatrix(E)
T̄ ← normalize(T)
L ← [0, 0, ...., 0]^tr
      {initialize L as a zero vector}

for iter = 1 to maxIter do
   L(s) ← L(s) + 1
            {Pump dye from the seed node}
   L ← L / ∑L(i)
            {Normalize dye level vector}
   L <- T̄L
{Distribute dye in network for one iteration}

end for
Output L
```

그림 2.2 Dye-pumping algorithm

위 알고리즘 수행 후 seed node와 P2P 연결 가능성이 높은 노드는 L(i) 값이 높게 나오며 그렇지 않은 노드는 L(i) 값이 낮게 계산된다.

III. Probabilistic Dye-pumping Algorithm

개선된 알고리즘은 mutual contact 노드와 내부 네트워크가 통신하였을 때, 그 패킷을 분석하였다는 가정에서 사용한다. 이 방법을 적용시키기 위해 행렬 T의 정의를 수정해야 한다. Dye-pumping에서 행렬 T가 mutual contact의 개수를 나타내었다면 Probabilistic Dye-pumping algorithm에서는 mutual contact 노드가 통신한 패킷이 봇에 의해 송신 되었을 확률의 합을 의미한다.

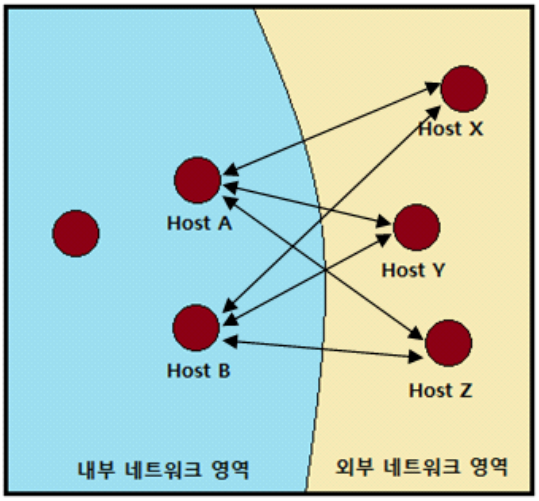

그림 3.1 Probabilistic Dye-pumping

그림 3.1에서 내부 네트워크 A, B는 mutual contact를 3개 가지고 있다. 네트워크 관리자는 경계 라우터에서 패킷을 분석하여 각 edge가 봇들의 통신 경로일 확률을 계산한다(P_{AX}, P_{AY}, P_{AZ}, P_{BX}, P_{BY}, P_{BZ}). 확률 P는 0과 1사이의 값을 갖는다. T(a,b)의 값은 다음과 같이 정해진다.

$$T(a,b) = P_{AX} \times P_{BX} + P_{AY} \times P_{BY} + P_{AZ} \times P_{BZ}$$

일반적으로 임의의 내부 네트워크 노드 i, j에 대하여 T(i, j)의 값은 다음과 같다.

$$T(i,j) = \sum_{n=1}^{M} P_{in} \times P_{nj}$$

(M : 노드 i와 j의 mutual contact 개수)

새로 계산된 T값을 적용시킨 L(i)의 계산을 위해서는, Dye-Pumping 알고리즘의 일부분을 수정해야 한다. 아래 알고리즘에서 수정된 부분은 기울임꼴로 표시하였다.

Probabilistic_Dye_Pumping(E, s, maxIter)

T ← computeProbabilisticMatrix(E)

$\overline{\mathbf{T}}$ ← normalize(**T**)

L ← [0, 0, …., 0]

{initialize **L** as a zero vector}

for iter = 1 to maxIter **do**

L(s) ← L(s) + 1

{Pump dye from the seed node}

$L \leftarrow \frac{L}{\sum L(i)}$

{Normalize dye level vector}

$L \leftarrow L\overline{T}$

{Distribute dye in network for one iteration}

end for

Output L

그림 3.2 Probabilistic Dye-pumping Algorithm

개선된 알고리즘에서는 기존의 알고리즘에서 L(i)값의 계산 순서를 바꾸었다. 이것은 L(i)가 의미하는 바가 다르게 된다.

그림 3.3은 내부 네트워크에서 node 1과 나머지 4개의 node가 mutual contact를 갖는 것을 나타낸 그림이다. 기존의 Dye-pumping 알고리즘에서는 L(1)의 값이 정해질 때, 행렬 T의 정규화의 영향으로 만약 node 2로부터 dye를 적게 받으면 node 3, 4, 5로부터 dye를 많이 받게 된다. 이 계산과정을 수정하지 않고 Probabilistic Dye-pumping에서 정의한 행렬 T를 적용시키면 다음과 같은 문제가 발생한다. node 1과 node 2가 감염되지 않은 노드이고, node 3, 4, 5중 감염된 노드가 존재하게 되면 node 2는 감염되지 않은 노드인데도 Dye-pumping 때보다 L(1)의 값이 다른 노드들에 비해 상대적으로 높은 결과를 얻는다. Node 1과 node 2가 감염되지 않으면 T(1, 2)의 값이 Dye-Pumping 때보다 Probabilistic Dye-Pumping에서 값이 작아지기 때문이다. 본 연구에서는 이런 경우를 피하기 위해 L값의 계산 과정을 수정했다.

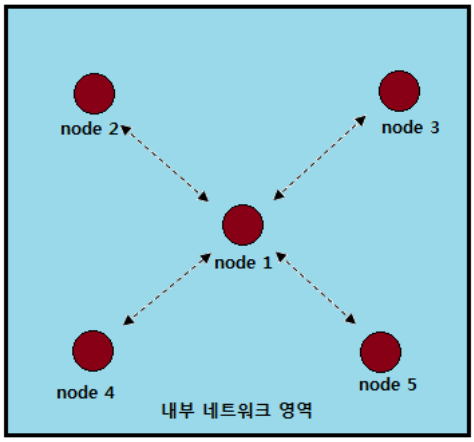

그림 3.3 내부 네트워크 내의 Mutual Contact

Ⅳ. 분석 결과

Ⅳ-Ⅰ 실험 환경

본 논문의 실험 환경 구축을 위해 대용량의 PCAP파일을 이용하여 약 38,000개의 내부 네트워크 패킷 정보를 수집하였다. 수집 된 정보는 C 언어를 이용하여 node와 edge를 생성하는데 이용하였다.

전체 노드 중 10%를 무작위로 뽑아 감염되었다고 설정하여 P2P 봇넷을 만들어주었고, P2P 통신 알고리즘인 Kademlia 알고리즘[5]-[6]을 이용하여 이들 사이에 edge를 만들어 주었다.

Ⅳ-Ⅱ 분석 방법

4.1절에서 논의한 실험 환경에서 Dye-Pumping 알고리즘과 Probabilistic Dye-Pumping 알고리즘의 수행 결과를 비교하기 위해 알고리즘의 성능을 비교 할 척도가 필요하다. 본 논문에서는 해당 알고리즘이 얼마나 많은 봇을 탐지 했는지 나타내는 값인 검출도와, 해당 알고리즘이 어떤 노드를 봇 이라고 판단했을 때, 그 판단의 정확성을 나타내는 값인 정확도를 정의하였다.

$$\text{검출도} : \frac{\text{내부 노드 중 L 값이 threshold 이상인 감염된 노드 개수}}{\text{내부 노드 중 감염된 노드 개수}}$$

$$\text{정확도} : \frac{\text{내부 노드 중 L 값이 threshold 이상인 감염된 노드 개수}}{\text{내부 노드 중 L 값이 threshold 이상인 노드 개수}}$$

각 알고리즘을 100번 수행한 뒤 검출도와 정확도의 평균치를 계산하여 Probabilistic Dye-Pumping 알고리즘의 개선도를 측정한다.

Ⅳ-Ⅲ 비교 결과

본 실험에서 threshold값을 높이면 검출도는 낮아지지만 정확도는 올라가고, threshold값을 낮추면 검출도는 높아지지만 정확도는 내려간다. 표 4.3.1은 두 알고리즘의 threshold값을 다르게 하여 비슷한 검출도가 나왔을 때 정확도의 개선 정도를 나타내는 표이다. 실험 결과, Dye-Pumping 알고리즘과 비교했을 때 Probabilistic Dye-Pumping 알고리즘의 정확도는 약 10%정도 높아졌다.

	Dye-Pumping (Th=0.002)		Probabilistic Dye-Pumping (Th=0.001)	
case	검출도	정확도	검출도	정확도
1	93.15%	94.44%	90.41%	100.00%
2	54.47%	100.00%	65.04%	100.00%
3	62.07%	80.36%	71.03%	100.00%
4	53.44%	80.46%	61.07%	94.12%
5	70.34%	93.58%	63.45%	100.00%
...	...		...	
96	66.90%	93.14%	73.24%	100.00%
97	45.93%	60.78%	61.48%	98.81%
98	76.69%	87.93%	78.20%	99.05%
99	86.29%	97.27%	75.00%	100.00%
100	49.58%	84.29%	63.03%	97.40%
평균	67.06%	89.49%	70.24%	98.55%

표 4.3.1 알고리즘 수행 결과

Ⅴ. 결 론

P2P 봇넷이 진화하면서 각종 탐지 시스템을 회피하는 기술이 발전하고 있다. 기존의 Centralized 봇넷을 탐지 및 완화할 수 있는 대응기술은 많이 연구가 되었으나P2P 봇넷을 탐지하는 대응기술은 꾸준한 연구가 필요한 실정이다.

본 논문에서는 P2P 봇넷에 속한 호스트들을 찾는 Dye-Pumping 알고리즘을 살펴보고, 여기에 확률을 적용하여 Dye-Pumping 알고리즘을 개선한 결과를 기술하였다. 본 논문에서 소개한 Probabilistic Dye-Pumping 알고리즘이 P2P 봇넷을 탐지할 수 있는 기술 마련의 기초로써 사용될 수 있을 것으로 기대된다.

참고문헌

[1] 전용희, 오진태, "봇넷 분류법 및 진화된 봇넷 구조", 정보보호학회지, 제18권, 제4호, pp. 76-86, 2008.8

[2] Baris Coskun, Sven Dietrich, Nasir Memon, "Friends of An Enemy: Identify Local Members of Peer-to-Peer Botnets Using Mutual Contact", ACSAC '10 Proceedings of the 26th Annual Computer Security Applications Conference, pp. 131-140, 2010

[3] Dan Liu Dan Liu, Yichao Li Yichao Li, Yue Hu, Yue Hu and Zongwen Liang Zongwen Liang, "A P2P-Botnet Detection Model and Algorithms Based on Network Streams Analysis", Future Information Technology and Management Engineering (FITME), 2010 International Conference, pp. 55-58, Oct. 2010

[4] Zhiyoung Huang, Xiaoping Zeng, Yong Liu, "Detecting and blocking P2P botnets through contact tracing chains", International Journal of Internet Protocol Technology, Vol. 5, pp. 44-54, Apr. 2010

[5] Guenther Starnberger, Christopher Kruegel, Engin Kirda, "Overbot: A Botnet protocol based on Kademlia", SecureComm '08Proceedings of the 4th international conference on Security and privacy in communication networks, Article No. 13, 2008

[6] Moritz Steiner, Damiano Carra, Ernst W. Biersack, "Fast Content Access in KAD", P2P '08 Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing, pp. 195-204, Sep. 2008