Automated Worm Fingerprinting

Sumeet Singh, Cristian Estan, George Varghese and Stefan Savage
Department of Computer Science and Engineering University of California, San Diego

HY558 - Spring Semester 2019

Rigaki Anastasia A.M 1100

rigaki@csd.uoc.gr

Abstract

First of all, in this paper it is mentioned that the network worms are a clear and growing threat to the security of today’s Internet-connected hosts and networks. Due to the Internet’s unrestricted connectivity and widespread software homogeneity, it allows network pathogens to exploit tremendous parallelism in their propagation. Thus, their solution to this problem, is an automated system for quickly detecting previously unknown worms and viruses based on two key behavioral characteristics called Earlybird. This system uses the method of “content sifting”. This implementation in combination with existing and novel algorithms has low memory and CPU requirements

Introduction

It is stated that since Code Red worm that took over 14 hours to infect its vulnerable population in 2001 and the Slammer worm that did the same in under 10 minutes the ability to defend against these outbreaks is extremely poor and has not advanced significantly. Here Earlybird is taking place.

EarlyBird automatically detects and contains new worms on the network using precise signatures. It generates signatures for all worms/viruses already known in network as well as for several new ones. The approach of content sifting, is based on two observations; the first, that some portion of the content in existing worms is invariant and second, that the spreading dynamics of a worm is atypical of Internet applications. It is rare to observe the same string recurring within packets sent from many sources to many destinations. By sifting through network traffic for content strings that are both frequently repeated and widely dispersed, it can be automatically identified new worms and their precise signatures.

Worm Detection

Many existing worms, excepting email viruses, select targets using a random process. Thus, a worm may will be highly unusual in the number, frequency and distribution of addresses that it scans. This paper describes some methods that already exist for worm detection and their limitations. The first one is scan detection. This method uses network telescopes. Network telescopes are passive network monitors that observe large ranges of unused, yet routable, address space. This paper assumes that worms will select target victims at random. So, a new worm will scan a given network telescope with a probability directly proportional to the worm’s scan rate and the network telescope’s size, which is the number of IP addresses monitored. As a result, large network telescopes will be able to detect fast spreading worms of this type fairly quickly. But, there are two limitations in this method. Firstly, it is not well suited to worms
that are not spreading randomly (e-mail viruses, worms spread via peer-to-peer applications). Secondly, scan detection can only provide the IP address of infected sites, not a signature identifying their behavior. The second method that is presented is called honeypots. Honeypots monitor idle hosts with untreated vulnerabilities. Any outside interaction with the host is, by definition, unsolicited and any malicious actions can be observed directly. Also, since the honeypot host is directly controlled, malicious code can be differentiated from the default configuration. In this manner, the “body” of a worm can be isolated and then analyzed to extract a signature. However, there also some drawbacks. They require a significant amount of slow manual analysis and they depend on the honeypot being quickly infected by a new worm.

The third method is host-based behavioral detection. Such systems dynamically analyze the patterns of system calls for anomalous activity, indicating code injection or propagation. But, they can be expensive to manage and deploy ubiquitously. Moreover, end-host systems can, by definition, only detect an attack against a single host and not infer the presence of a large-scale outbreak.

Finding worm signatures

This paper is making some extra assumptions as for the network traffic, such as, that network worms must generate significant traffic to spread and that this traffic will contain common substrings and will be directed between a variety of different sources and destinations. Moreover, they assume that identifying the traffic pattern is sufficient for detecting worms.

Estimating content prevalence

According to the paper, content prevalence is the key metric for identifying potential worm signatures. Identifying common content involves finding the packet payloads that appear at least x times among the N packets sent during a given interval. However, a table indexed by payload can quickly consume huge amounts of memory. Memory consumption can be reduced considerably by indexing the table using a fixed size hash of the packet payload instead of the full payload. After a certain hash value has repeated x – 1 times, the next packet with this hash is reported. In the absence of collisions, the associated content will have appeared exactly x times. By selecting a hash function with suitably large range (e.g., 32 or 64 bits) the collision probability can be minimized. For the problem of memory-footprint, they use multi-stage filters. A piece of content is hashed using hash function h1 into a Stage 1 table, h2 into a Stage 2 table, etc. and each table entry contains a counter that is incremented. If all the hashed counters are above the prevalence threshold, then the content string is saved for address dispersion measurements.

Estimating address dispersion

Address dispersion is critical for avoiding false positives. Without this additional test a system could not distinguish between a worm and a piece of content that frequently occurs between two computers. To quantify address dispersion, the distinct source IP addresses must be counted and destination IP addresses associated with each piece of content suspected of being generated by a worm. In this paper, the first approach that has been followed is direct bitmap. Each content source is hashed to a bitmap, the corresponding bit is set, and an alarm is raised when the number of bits set exceeds a threshold. However, direct bitmap loses the ability to estimate the actual values of each counter.
Instead, they invented a counting algorithm that leverages the fact that address dispersion continuously increases during an outbreak. Using this observation, they devise a new, compact data structure, called a scaled bitmap.

```plaintext
UpdateBitmap(IP)
1. code = Hash(IP)
2. level = CountLeadingZeros(code)
3. bitsize = FirstBits(code « (level+1))
4. if level > base and level < base+numbmps
5. SetBit(bitcode,bitmap[level-base])
6. if (level == base and CountBitsSet(bitmap[0]) == max)
7. NextConfiguration()
8. endif
9. endif
```

Scaled bitmap uses `numbmps` bitmaps of size `b` bits each. The bitmaps cover progressively smaller portions of the hash space. When the bitmap covering the largest portion of the hash space gets too full to be accurate (See first figure previously). Moreover, they use the method of “recycling” the bitmap (See second figure). To compute an estimate of the number of distinct IP addresses, they multiply an estimate of the number of addresses that mapped to the bitmaps by the inverse of the fraction of the hash space covered by the bitmaps. A correction is added to the result to account for the IP addresses that were active in earlier configurations, while the current bitmaps were not in use at their present levels.

CPU scaling

The previous methods of multi-stage filters for detection of content prevalence and scaled bitmaps to estimate address dispersion, decreases memory usage and limits the amount of processing. However, each payload string still requires significant processing. In this paper, they use value sampling and select only those substrings for which the fingerprint matches a certain pattern (e.g. the last 6 bits of the fingerprint are 0). Consequently, the algorithm will systematically ignore some substrings, but track all occurrences of others.

Putting it together

As each packet arrives, its content (or substrings of its content) is hashed and appended with the protocol identifier and destination port to produce a content hash code. The resulting hash codes are used to index the address dispersion table. If an entry already exists, then the address dispersion table entries for source and destination IP addresses are updated. If the source and destination counts exceed the dispersion threshold, then the content string is reported. They use four independent hash functions of the content hash to create 4 indexes into four counter arrays. Using the conservative update optimization, only the smallest among the four counters is incremented. If all four counters are greater than the prevalence threshold, then a new entry is made in the address dispersion table – with high probability, the content has appeared frequently enough to be a candidate worm signature.
System design

The EarlyBird system consists of two major components: Sensors and an Aggregator. Each sensor sifts through traffic on configurable address space “zones” of responsibility and reports anomalous signatures. The aggregator coordinates real-time updates from the sensors, coalesces related signatures, activates any network-level or host level blocking services and is responsible for administrative reporting and control.

Performance

As for the processing time for measurement of the overhead, they instrumented the interface of each component to count elapsed CPU cycles. Using value sampling, the processing overhead dropped from 0.409 msec to 0.042msec. They found out with the use of multistage filters with 4 stages, with each stage containing 524288 bins, that the core EarlyBird function currently consumes less than 4 Mbytes of memory.

Traced-based verification

They found two principal sources of false positives: common protocol headers and unsolicited bulk email (SPAM). In the former category, over 99 percent of all false positives result from distinct SMTP header strings or HTTP user-agent or content-type strings. The other principal source of false positives are SPAM e-mails which can exceed address dispersion thresholds due to the use of distributed mailers and mail relays. Moreover, popular files distributed by the BitTorrent peer-to-peer system can satisfy the content prevalence and address dispersion criteria during their peak periods of popularity. BitTorrent’s file striping creates a many-to-many download profile that mimics that of a worm. Due to uncontrolled environment that the experiments have took place, they could not quantitatively demonstrate the absence of false negatives.

Live experience with EarlyBird

Earlybird has also detected precise signatures for variants of CodeRed, the MyDoom mail worm and most recently for the Sasser, and Kibvu.B worm.