Worm Identification

Horace Heffner January, 2001

Internet Service Provider's (ISP's) have the first opportunity for worm identification, finding key stings in a new worm which can later be used by virus and worm removal software for identification and removal of the offenders. The purpose here is to suggest a specific early worm identification method (EWIM) which involves both automated and (unfortunately) some manual procedures. The suggested identification software can be located in email servers, but also possibly routers.

The identification of high frequency stings of data is a procedure used in various data compression algorithms. The basis of this EWIM is to identify suddenly commonplace strings (SCS's). SCS's are strings that have not been identified as commonplace before, that suddenly have a high frequency. Additionally, the method is intended to identify multiple SCS's having a high joint frequency, i.e new clusters. That is to say a group of stings having both a suddenly high frequency and which also have a high frequency of occurrence in the same message, i.e. a clustering.

The SCS cluster identification algorithm consists of six general steps:

1. Removal of known spam resulting in residual email

2. Elimination of normally common strings

3. Identification of high frequency strings in residual email

4. Building of SCS coincidence matrix.

5. Periodically building a potential worm identifier list and resetting

6. Building a sample set of emails having the 10 most common newly identified string clusters.

Step 1, the removal of known spam and identified worms can be done using a filter which recognizes a concurrence of strings, a cluster or clusters that identify an email as known common benign spam. Routines for fast recognition of concurrent strings in a given data string are readily available to those skilled in the art of programming. The string combination specifiers for this step could fairly quickly be built manually initially. The email that passes this first step is called residual email.

Step 2, the elimination of known high frequency strings, is accomplished using a dictionary of commonplace strings. All common HTML strings would be eliminated by this step, for example. This dictionary can initially be built by simply accumulating the highest frequency stings observed in email servers around the world, or just at a given installation and server. In this step high frequency strings that are threatening can be preserved by checking a dictionary of such strings prior to discarding the string in question.

Step 3, the identification of high frequency strings in the residual email, can be done over a periodic time interval by building a dictionary of strings encountered in the residual email and their frequencies. When the space in the high frequency string dictionary being built is full, the low frequency half of the dictionary can be purged. This process could easily build a dictionary of the most common 10,000 strings in the residual email. The process of identifying high frequency strings in given input strings is used commonly in data compression algorithms. It is computationally

intensive, and compute time depends on the maximum possible length of strings identified. The computational time and memory requirements for this step may indicate a need for an essentially one way link from the mail server machine to a machine dedicated to this process.

Step 4 consists of the building of the SCS coincidence matrix by assigning each string in the dictionary created in step 3 a unique number, an identifier. This identifier is then used to subscript into a two dimensional incidence matrix that keeps track of the count of occurrences of string j given that string i is in a residual email. A string list of 10,000 strings results in an incidence matrix of about 1.6 Gig. As high frequency pairs are identified, these can be placed in a special "string of strings" dictionary and each entry there also identified by a unique identifier, an identifier not used by any other string or string of strings. When the string of strings dictionary is not empty Step 3 must also find any relevant identifiers from the string of strings dictionary as well.

Step 5 consists of, after some time interval, say 1 to 6 hours, selecting the 10 (or so) most common clusters and from them building a "worm identifier" consisting of text commands with data that can be used by a program or person to specify a set of strings comprising a worm identifying cluster.

Step 6 consists of collecting samples of emails having the identified clusters for manual perusal. If the new clusters are determined to be infectious, then the worm identifier command can be posted to a website for distribution.

This method could be applied within routers, but the technique would be somewhat limited in scope because a given router may see only parts of a given email. However, since only a few routes pass much of the email in the USA, it may be possible to obtain sufficient email samples from the routers to apply the process, which is in fact merely a sampling process anyway.

It is also noteworthy that that identifier could be further tagged or extended to include the source of the string, i.e. a given header field, the email proper, or by attachment type.

This method is far from perfect or automatic, but it gives a good starting place for building an early warning system for worms. The refresh time period can be long or short depending on mail volume and the availability of personnel for reviewing or testing the culled findings.