Multi-agent anomaly-based APT detection

Wim Mees
Royal Military Academy, Department CISS
Renaissancelaan 30, 1000 Brussel, Belgium

wim.mees@rma.ac.be

ABSTRACT

Protecting corporate networks against persistent malware infections is becoming an increasingly difficult challenge. Most networks will suffer from an infection sooner or later, and when this happens, it is very important to identify the compromised host as quickly as possible before any real damage is done. Therefore we focus in this paper on the detection of the command & control channel between a malware and an outside server.

We will discuss an approach for detecting this C^2 channel using a multi-agent aggregation of evidence method that combines inputs from anomaly and signature-based techniques. Two anomaly-based agents will be presented in more detail, one for detecting suspicious HTTP transactions, and one for detecting suspicious DNS requests.

Finally, a number of results, obtained with these agents, will be shown.

1.0 INTRODUCTION

The security of corporate networks relies to a large extent on the ”hard shell - soft interior” model, with a tightly controlled interconnection choke point with the external world where a security policy is enforced. The currently available security controls to implement this model, however, mainly focus on protecting against known attacks. As a consequence, they have unfortunately proven to be unable to protect against many so-called zero-day attacks, not in the least because the human operator remains a weak link that cannot entirely be compensated for by technical measures. We have indeed seen a number of attacks, where a malware was successfully injected into a network through for instance a spearfishing attack, and was then used as a base for slowly attacking other resources in the network over longer periods of time. These types of attacks are sometimes referred to as *”advanced persistent threats”* (APT).

Given the fact that one can assume that sooner or later an infection will occur, and given the ”soft interior” model that allows a successful attacker to hop from a low profile infected host to more interesting targets more easily, an important conclusion in a risk management process is that it is essential to be able to detect as soon as possible when a host in the corporate network gets infected and starts being controlled by an outside attacker, even when the compromised host itself contains only low value information.

In order to satisfy the information needs of the users, a number of basic services, such as email and surfing the web, are typically available in corporate networks to nearly all the hosts. Even when inspected in relays and proxies at the entry point to the corporate network, these services are still a vector through which many users inflict malware onto their own systems. Furthermore, they are also the preferred channels for many malwares to tunnel their communications with the outside world.

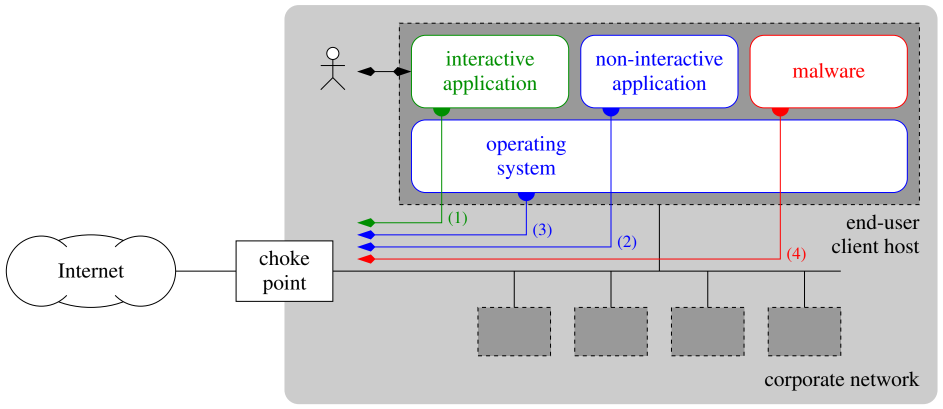

Figure 1: flows originating from an end-user host

Consider a host inside the corporate network, as depicted in figure 1. It can generate different types of packet exchanges with the outside world: (1) exchanges related to a user that interacts with an application, for instance while surfing the web using a browser; (2) exchanges caused by a non-interactive application that automatically, typically at fixed intervals, contacts an outside server, for instance to verify for software or information updates; (3) exchanges initiated by the operating system, again typically at fixed intervals, for instance to look for software updates or to synchronize clocks; and finally (4) exchanges originating from a malware that tries to remain undetected but needs to initiate connections to receive commands from a command & control server or to exfiltrate captured data.

The goal of our research is to build a tool that allows a network administrator to analyze the flows at the level of the choke point in an interactive way and assists him in separating the flows of type (4) from the background flows (1)-(2)-(3). For the purpose of this paper, we consider that at the level of the choke point three protocols are made available through application layer gateways:

hypertext transfer protocol (HTTP): for retrieving web pages or other types of content from a server or for posting data to a server.

simple mail transfer protocol (SMTP): for sending email to outside recipients.

domain name system (DNS): for translating hostnames into numerical IP addresses.

All three protocols can be used by a malware for tunneling a command & control channel to a C^2 server on the Internet and therefore will be subject to inspection.

In this paper we will first present the overall inspection approach in section 2.0. In section 3.0 two algorithms for detecting suspicious traffic will be discussed in more detail. Some examples of anomalies that were detected using these algorithms are subsequently presented in section 4.0. Finally, in section 5.0, we will present our conclusions.

2.0 DETECTING SUSPICIOUS TRANSACTIONS

The goal of our system is to combine different types of information about the monitored data exchanges in order to rate their degree of suspiciousness and aggregate them for each host in the private network. A human operator

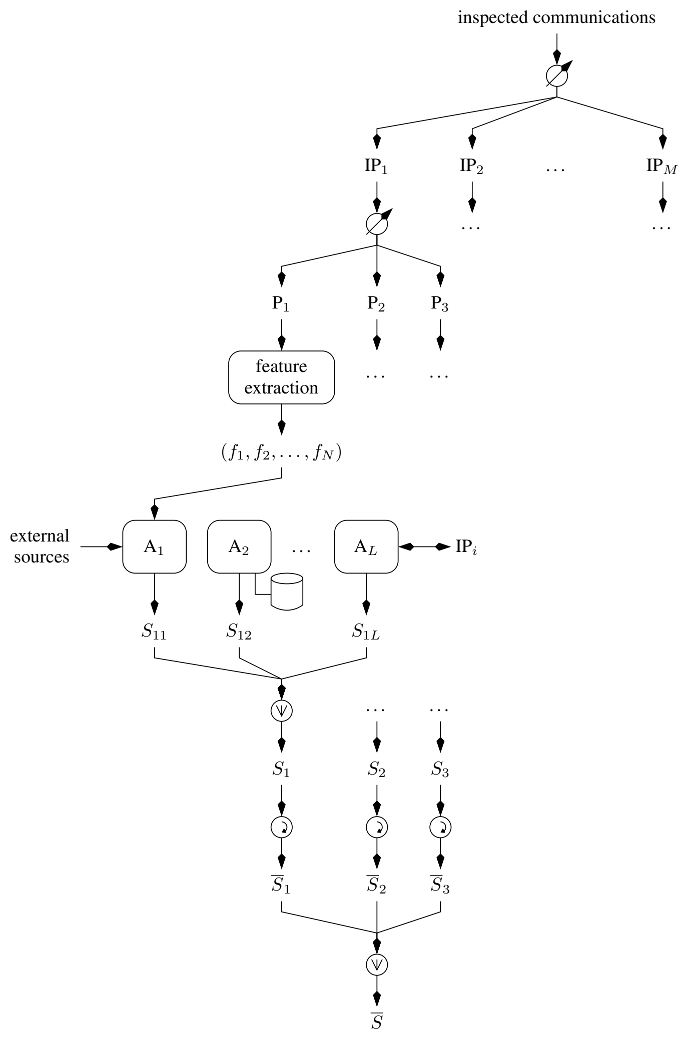

Figure 2: feature extraction, evaluation and fusion

can then inspect in an interactive way the information related to those hosts that get the highest aggregated suspiciousness levels. Figure 2 shows the processing architecture of the system.

First the communications[1] are sorted by IP address of the internal host that is involved. This is required since we need to look at the behavior of individual hosts in order to detect those that behave differently from the others and from what can generally be expected from them.

The next step consists in separating the different protocols P_i we consider. For the moment this is HTTP, SMTP and DNS. We then extract a protocol-specific feature vector $(f_1, f_2, \ldots, f_N)$ from each communication, containing for instance the external IP address that is involved, the timestamp of the communication, the URL

[1]The term communication is used in this paper to denote an exchange between an internal client and an external server and consists of a client emitted request followed by a reply from the server.

Such a communication can correspond with a TCP connection, but there may also be several communications inside one TCP connection, for instance in the case of HTTP where the same TCP connection can be reused for several HTTP request/reply exchanges. In the case of DNS on the other hand, a communication consists typically of a UDP datagram from the client containing a request, followed by a UDP datagram from the server with the reply.

that was requested or the hostname that was resolved.

The feature vector is submitted to a limited number of high-level protocol-specific agents, that independently evaluate its degree[2] of suspiciousness. Each agent applies a different subset of the domain knowledge that describes the difference in behavior between malware generated communications and regular communications.

In the example of figure 2, agent A_1 will consult a number of external sources to evaluate the reputation of the outside host with which the communication occurred, using an approach that is based on the methods proposed by Antonakakis [1] in the case of the DNS protocol.

Agent A_2 compares this communication with previous ones for the same internal host and protocol, stored in a database, in order to distinguish ”robotic” from ”human behavior”. For email messages for instance we have built an agent based on the work of Gianvecchio and Wang [2] that incorporates the domain knowledge that human behavior is more complex than bot behavior, and therefore is characterized by higher entropy measurements. We have furthermore built a DNS agent that tries to determine whether the domain names or hostnames in a set of requests where automatically generated by a computer rather than chosen by a human being.

Finally, agent A_L will compare the communication with the communications performed by other internal hosts in the same subnet, as well as with hosts in other subnets. This is a translation of the domain knowledge that users working in the same department, and thus belonging to the same VLAN and subnet, tend to have similar tasks and professional interests, and therefore tend to generate similar network flows. The same similarity requirement can, albeit to a more limited extent, be applied to hosts in other subnets.

The degrees of suspiciousness S_{ij} from the different agents are aggregated into a single degree of suspiciousness S_i for the given communication using Yager’s ”ordered weighted averaging” (OWA) operator [3]. This aggregation operator was chosen because it allows us to express in an easy way how many agents must agree before reaching a consensus and enables us furthermore to select an appropriate weight distribution between them.

The next steps consist in aggregating over subsequent communications of the same protocol, resulting in a $\overline{S}_i$ per protocol, and finally in aggregating over the different protocols in order to obtain a global degree $\overline{S}$ for the host with IP address IP_1. Based on this global degree for each IP address the security analyst can then decide which hosts to examine in more detail.

3.0 AGENTS

3.1 Statistical hypothesis testing for detecting automatically generated DNS entries

In the case of DNS communications we want to verify whether the domainnames or hostnames that are queried were generated automatically by a software using a random number generator or whether they ware rather chosen by a human being.

Malware writers tend to use constantly changing domain- and hostnames for establishing a rendez-vous point between an infected PC on the corporate network and a C^2 server on the Internet. Since these names are generated in a pseudo-random fashion, often using the standard uniform random number generator available

[2]We use the expression ”degree of” because our agents produce a degree of membership to the fuzzy set of ”suspicious communications”.

Some of them internally use domain knowledge that is expressed as a mini fuzzy expert system and the inference hereof automatically produces a membership degree. Others produce a single crisp value that is then mapped into a membership degree by using a single fuzzy set that expresses the expert’s knowledge on how this crisp value shall be interpreted.

in most programming languages as an input, this agent adopts the hypothesis that the generated sequences of symbols will also present a uniform distribution over all the symbols that are used.

First we build the DNS tree for all the hostname requests that were sent by the internal hosts. Then we examine each node of the tree separately. If it has a sufficient number of children, we sort the children in bins based on their length. This is necessary because we may have at the same level a combination of computer generated children and regular children. The regular children may pollute the hypothesis test and cause the test to fail. Since the computer generated names are typically of constant length, we separate the names over bins based on length, and suppose that the few regular names that share a bin with the computer generated ones will not pollute their distribution in a significant way.

If such a bin contains a sufficient number of entries, we concatenate all the names, normalize the resulting sequence by removing everything which is not a letter and converting the letters to uppercase, and finally map the letters 'A' to 'Z' onto the values 0 to 25. We now have to verify whether the distribution of these values is uniform or not.

The hypothesis of a uniform distribution is evaluated using a progression of tests. This way no unnecessary computational resources are spent on sequences that are far from being uniformly distributed.

First we test the mean value and the variance of the observed sequence and compare them with the values we should have obtained for a perfectly uniform distribution. Only if they are within an acceptable range of the reference values, the final hypothesis test is performed. Otherwise the hypothesis is declared invalid and the agent returns a zero degree of suspiciousness evidence.

The n observed values are now distributed over k bins, where the number of bins is chosen at 26 (or lower when needed in order to have at least 5 reference samples per bin). We consider that the n values are independent observations, which means that the outcome of one observation has absolutely no influence on the outcome of the other observations. Notice that this is true for the computer generated domainnames we are interested in but not for the manually chosen names. The number of times value i occurs in the sequence, or in other words the number of elements in bin i, is called Y_i. For a uniform distribution, the expected probability p_i for value i is equal to $1/k$.

The χ^2 statistic of the observed quantities Y_0 to Y_{25} is then defined as follows.

$$V = \sum_{i=1}^{k} \frac{(Y_i - np_i)^2}{np_i} \tag{1}$$

For the sequences that do not follow a uniform distribution at all, the value of V will be high and there is no confusion possible. For sequences that are generated by a computer algorithm based on a uniformly distributed random number generator however, the values of V will be most of the time rather low. However, we only have one sample sequence and cannot simply throw the dice a couple of times more to make sure, and furthermore, what is meant by "rather low"?

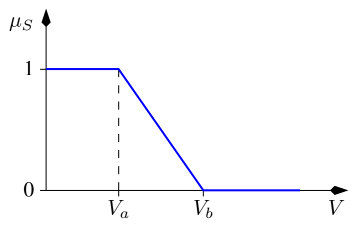

Figure 3: membership function for S as a function of V

We handle this vagueness through a fuzzy set S, representing the ”suspiciousness as judged by this agent”, and define its membership function μ_S as a function of V, as is shown in figure 3.

The values of the transition points V_a and V_b are chosen from the table in ”The Art of Computer Programming” by Knuth [4] for a number of ”degrees of freedom” $\eta = k - 1 = 25$ and in such a way that 50% of the samples will have a membership degree of 1 and that 95% have a membership degree of at least 0.5. For our experiments we use $V_a = 25$ and $V_b = 50$.

3.2 Frequency analysis for detecting periodic HTTP requests

Malware that uses an HTTP command & control channel will periodically issue an HTTP or HTTPS request to a external webserver. The difference between a software agent and a human user that revisit periodically the same website, is that the time between two visits will be fixed in the case of program, be it malignant or benign, but will vary for the human user.

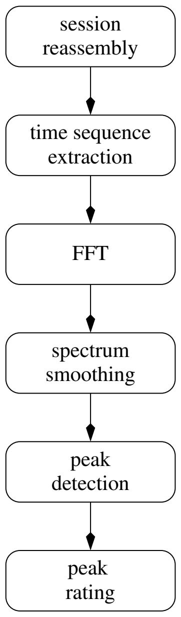

Figure 4: frequency analysis of HTTP Requests

The detection scheme for distinguishing between computer and human behavior is shown in figure 4. The first step consists in grouping related requests into sessions, based on the similarity between URLs, the time difference between requests, and the referrer and user agent information in the HTTP header if this is available.

The next step groups sets with identical sessions together and computes the time sequence for each group that is sufficiently large. When there are only a few elements in a group, the corresponding time sequence is not suitable for further processing and therefore the group is removed.

Each time sequence is then sampled and wrapped around on a 24 hour grid, as well as on a 7 day grid. For each of these the spectrum is then computed using an FFT algorithm.

Because the obtained spectra are rather noisy, they are not suitable yet for peak detection. Therefore we first apply a smoothing filter to detect the envelope of the spectrum. This step uses a combination of an order statistic filter and a regular low-pass filter.

Finally the peaks are detected and their amplitude is compared with the average level of the spectrum.

4.0 RESULTS

We will present in this section a few examples of the results obtained with the agents described in the previous section.

- detection of automatically generated domain names

  The DNS robot name detection agent has been applied to the DNS communications of a set of about 50 hosts that were recorded over a period of time of about 4 months, resulting in more than 5 million requests.

  Some of the results:

  - subdomains of *.edu* with 4 letters.

    For one host the following subdomains of *.edu* were present in the set of recorded DNS queries:

```
suny,vill,utsa,jalc,unca,risd,sans,yale,naic,bgsu,drew,iese,mscd,cwru,miis,
mtsu,umbc,csus,usna,csun,nmsu,uwsp,aast,uofs,afit,unmc,uwec,msmc,olin,fcps,
okbu,sfsu,neiu,kent,uncc,fhsu,umuc,uiuc,accd,nova,open,gcsu,hvcc,etsu,uwyo,
uccs,reed,apsu,ncsu,selu,cofc,salk,uprm,cccd,lstc,ucar,rice,hope,jhmi,rush,
ucop,lhup,nrao,ohio,tcnj,saic,unlv,edcc,pitt,cwru,poly,umkc,mstc,ucla,whoi,
usra,wcsu,cuny,case,cocc,erau,ship,kumc,ohsu,swri,fgcu,ncat,noao,rose,ualr,
msoe,mayo,utah,grin,udel,nvcc,sdsc,nwtc,ucsc,ucsb,ucsd,ucsf,uwgb,sdsu,gvsu,
iris,spsu,uark,duke,tamu,umsl,utep,wisc,ccsf,mnsu,aces,plts,orst,njit,sjsu,
usma,sals,mass,nasm
```

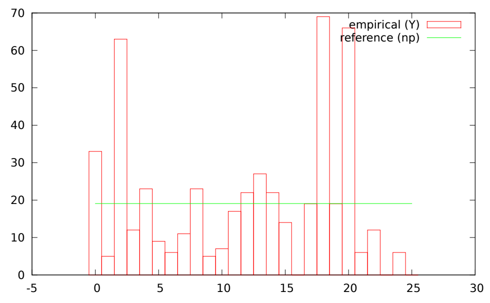

Figure 5: frequencies for 4-letter subdomains of .edu

The empirical and reference frequencies are shown in figure 5. The resulting value for V is 491, which is clearly too high to pass the test. The agent therefore concludes that the degree of suspiciousness is 0, which means that these names were clearly not generated by a computer program.

  - subdomains of *.dns.xx.fbcdn.net* with 4 letters.

    Observed names:

```
aklw6c1a,6240e5nx,2y908bez,b3v6duos,9g9q0hm6,cp4dbmdv,9whf61os,1iq691ua,
6gpj8rjs,3fnswxq5,bywuim7k,5lm1hxro,61mysxdi,8tajgj5r,b38pllth,72u4tfs0,
c0hp9xdk,7hh88yti,6svfo2p4,8etv7ppe,9hon79wi,68ebennm,cr2f7uoj,9dm8z9n5,
c3ozjng5,a7b0ub6a,bghtfc12,2s6ctufu,67mgrxby,c5voqbir,8qei91d2,3ddef2bt,
76nqmkdt,drrh8h27,binbk9b3,10rfs8d6,5ydqpq5k,bukvkagv,1o6q6nvb,4ul32mei,
9he29ozp,b8za4glb,cpv4vdfr,5c3strim
```

The corresponding letter frequencies are shown in figure 6. The value of V for this set is equal to 23, which is translated into a suspiciousness degree of 1, which means that the agent is convinced that these names were clearly generated by a computer.

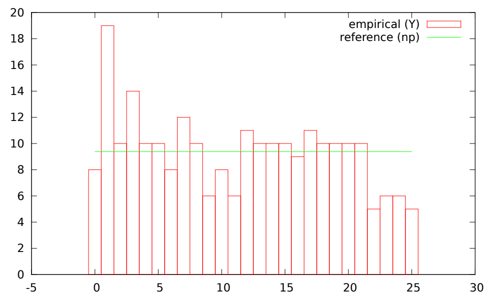

Figure 6: frequencies for 4-letter subdomains of .dns.xx.fbcdn.net

We found in our tests that the hard zero or one values for S correspond with the obviously machine generated or not machine generated cases, and that the fuzzy area $0 < S < 1$ encompasses the less obvious situations.

The last example also shows that there are situations where the names are automatically generated yet the requested domain names are not suspicious, resulting in false alerts. This is solved by white-listing.

- detection of machine-issued repetitive HTTP requests

  We have also built an agent that isolates HTTP requests that are repetitively issued by a software, possibly a malware, from those that are issued by a human user who is for instance using a browser.

  The agent computes the frequency spectrum of the requests and extracts peaks that are sufficiently strong. The agent will typically find a number of automatic updating services that are running on each host, for instance retrieving Microsoft's *"certificate revocation list"* (CRL):

  `http://crl.microsoft.com/pki/crl/products/CodeSignPCA2.crl`

  Figure 7 show the time sequence for this URL for a given host, as well as the spectrum (blue dots), and a smoothed version of the spectrum (red line). The peaks clearly show the fundamental frequency that corresponds with a periodicity of 12 hours, as well as its harmonics.

  The same agent also detected the following URL, with the corresponding frequency analysis shown in figure 8.

  `http://149.20.56.32/search?q=0`

  This URL was due to an infection with the Conficker worm.

5.0 CONCLUSIONS

We have presented in this paper an approach for automatically detecting hosts inside a corporate network that present potentially suspicious behavior. Behavior that might indicate the presence of a malware that is trying to connect to an external command & control server.

The domain knowledge that describes how a malware can be detected at the choke point between the corporate network and the external world is distributed over a number of agents, that independently form a judgment on each communication for each protocol and each host.

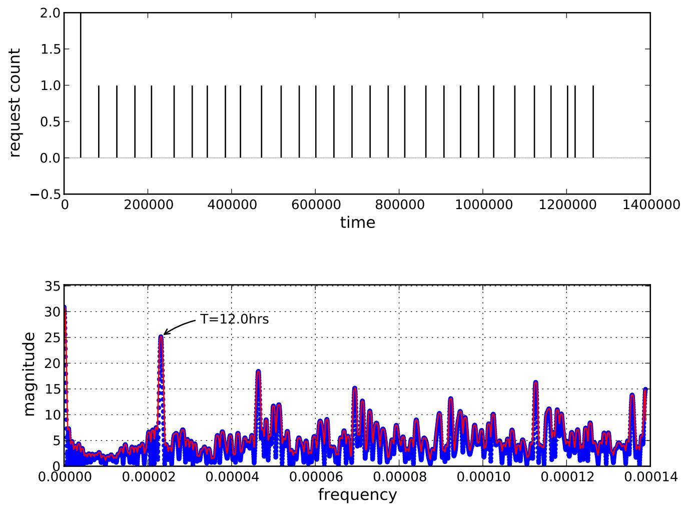

Figure 7: frequency analysis for the Microsoft CRL updates

These evaluations are combined using the ordered weighted averaging aggregation operator into a single suspiciousness level, that can then be used by a human analyst to select the hosts he wants to investigate further by looking in more detail at their connections with the outside world.

We have presented two of the anomaly-based agents that are being used, one that detects automatically generated host of domain names, and a second one that looks for repetitive HTTP requests.

We have finally also presented experimental results where these agents were indeed able to detect ”robotic” behavior, that after further analysis in some cases indeed proved to be caused by C^2 traffic from malware infected hosts.

REFERENCES

[1] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster: Building a dynamic reputation system for DNS. In: Proceedings of the 19th USENIX conference on Security (USENIX Security 2010). USENIX Association, Berkeley, CA, USA.

[2] S. Gianvecchio and H. Wang: Detecting covert timing channels: An entropy-based approach. In: Proceedings of ACM CCS 2007, Alexandria, VA., USA, October 2007.

[3] R. R. Yager: On ordered weighted averaging aggregation operators in multicriteria decisionmaking. In: IEEE Trans. Syst. Man Cybern. 18, 1 (January 1988), 183-190.

[4] D. E. Knuth: The Art of Computer Programming Volumes 1-3 Boxed Set (2nd ed.). 1998. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

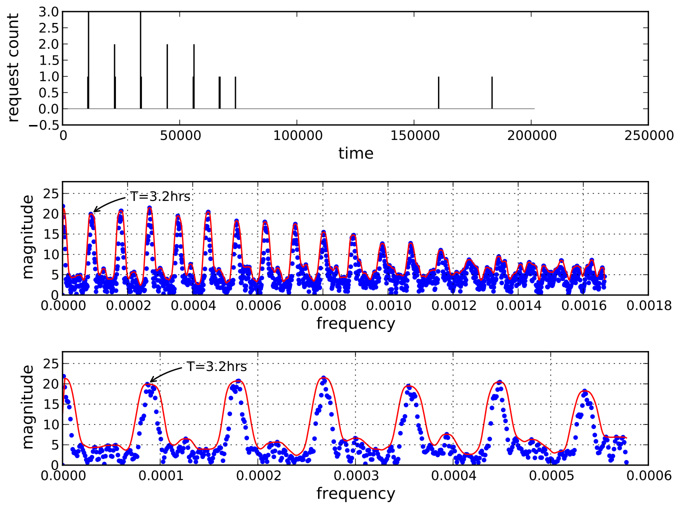

Figure 8: frequency analysis for the Conficker worm