Combatting Email Borne Pests using Open Source Tools

Joel Sing

Director, Ionix Technology

<joel@ionix.com.au>

ABSTRACT

Recently, the number of email borne pests has increased significantly, causing a flood of useless email to arrive in user mailboxes. This paper details the tools and techniques used by a small hosting provider, in order to combat these pests, namely virii and spam. The methodology used in designing the mail scanning system is detailed, including policy decisions made regarding quarantining, bouncing and notifications. Finally, the effectiveness of the implementation is discussed, along with details of possible future work.

1. Introduction

Over the last couple of years the number of viruses being transmitted via email has increased significantly, as has the number of Unsolicited Commercial Email (UCE) messages, more commonly known as spam. These email borne pests cause a flood of useless email to arrive in user mailboxes, wasting both time and money, severely impacting the usability of this communication medium.

This paper details the tools and techniques used by a small hosting provider, in order to combat these email borne pests. In the past a number of simple configuration settings have been used to reduce the amount of spam received. Whilst this proved to be suitable for a period of time, it has become ineffective as spammers became more desperate and began using a number of different techniques to circumvent these tools. Many of the techniques used by spammers in an attempt to avoid detection are discussed in section 2.2.

The methodology used in the design the mail scanning system, including policy decisions made regarding quarantining, bouncing and notifications, is detailed in section 4. An in-depth discussion regarding the selection of each tool is provided in section 5, with the implementation and configuration of the entire system being covered in section 5.2; information regarding the addition of greylisting[17] can be found in section 6. Finally, the effectiveness of this implementation is summarised in section 7, with details of possible future work being given in section 8.

2. Email Borne Pests

2.1 Virii

In recent years there has been a large increase in the number of viruses and worms that spread themselves via email, the Melissa virus released in 1999 arguably being the start of mass-mailing virii/worms. Email is a simple and effective means of transferring data between systems, in this case the data being executable content. In addition, a large number of security issues with Microsoft's email clients have allowed a number of viruses to spread and invoke themselves without the user doing anything more than simply viewing the message. Email borne viruses range from being virtually harmless, doing nothing more than spreading themselves, through to being potentially devastating.

2.2 Unsolicited Commercial Email

Simply put, Unsolicited Commercial Email (UCE) is email that you receive without ever having requested it, typically being of a commercial or advertising nature. UCE, more commonly known as spam, requires minimal effort and cost to send, allowing spammers to deliver millions of messages a day. Given the minimal costs involved, only a small conversion rate is required to make it worth their effort; unfortunately there are people who purchase a product or service that is advertised via spam.

Over the last couple of years spam has become increasingly difficult to identify using content and header analysis. Most spam is sent using a forged sender address, meaning that the message appears to be sent from someone other than the originator. This has the side effect of bouncing all rejected and undeliverable messages to the email address that appears to be the sender. In order to defeat spam identification

tools, particularly those that do content analysis, spammers munge content by intentionally misspelling words and substituting numbers for letters. Random words are often added to messages or even entire paragraphs of text from various sources, also in an attempt to bypass spam filters. It is worth noting that whilst nearly every part of a spam message can be modified or falsified, the most critical part from a spammers perspective, the URLs within the message, typically cannot.

To further complicate matters, spammers typically use open relays to deliver their messages. This has the affect of masking the original system from which the spam was sent, making it difficult to use IP based blocking. A number of the viruses/worms that have appeared recently have built in SMTP[18] relays that spammers use to their advantage - a number of sources suggest that there is a distinct connection between these viruses and a number of organisations who send spam for profit.

Another technique that spammers use in order to avoid detection is the delivery of email to secondary or tertiary mail exchangers, instead of delivering mail directly to the primary mail exchanger, even if it is available. This means that IP based blocking will not work for any email that is delivered via mail exchangers that are outside your control. In many situations a corporate level ISP will provide an offsite mail exchanger as part of the connectivity service; for obvious reasons the same tools used to combat email borne pests cannot be deployed on this system.

2.3 Reasons to Engage in Combat

There are a number of reasons to combat email borne pests, ranging from bandwidth usage to the prevention of virus transmission. From a corporate perspective, spam is extremely destructive. Huge amounts of time can be spent sorting through one's inbox to separate unwanted email from wanted email, which only adds insult to injury. The bandwidth used by both virii and spam will typically incur per megabyte charges from the upstream ISP and for those who do not have access to broadband and are either using a dialup or mobile connection, significant time can be wasted waiting for unwanted email to download.

In addition, the receipt of email messages that are transporting virii increase the possibility of infection, namely for our Microsoft based counterparts. For non-Microsoft based users, the bandwidth and time taken to download and delete each message adds up.

3. Techniques for Combatting Email Borne Pests

The first step in combatting email borne pests is identification, in other words separating the pests from the real email. The action that is taken following identification will depend on the form of identification used and the policies that are in place.

Initially, email borne pests were identified manually with sender address/domain and IP based blocklists being manually maintained. The effectiveness of these two blocklists combined was fairly high, although maintaining the lists was a time consuming and manual process. The primary downside of this approach is that one has to receive the spam before it can be added to the blocklist. Over the last year or two this process has become less and less effective as more and more messages are being sent with forged from addresses and via SMTP servers that are open relays or zombies.

The first line of defense is blacklisting via sender IP address. This allows for SMTP servers to reject the receipt of messages from specified senders or for the TCP connection to be denied altogether. Given that large amounts of spam originate form a small number of systems, the effectiveness of a well maintained blacklist, such as those provided by the Spamhaus SBL and XBL, can be extremely effective.

The second line of defense is blocking by envelope sender. Many SMTP servers can be configured to reject messages from specific email addresses or originating domains, Qmail[13] being no exception with its support for *badmailfrom*. It is worth noting that this is trivial to work around by changing the email address specified during the *MAIL* command of an SMTP session, however it is fairly effective against small time spammers, typically those companies that do their own unsolicited mass mailouts.

At this point, the email message has typically been received by the SMTP server, hence the bandwidth and associated SMTP processing costs have been incurred. Analysis of the actual email message can now be performed using a large number of different techniques and tools. These include virus scanning and various forms of content analysis. Depending on the outcomes of the analysis the message may be delivered to the intended recipient, tagged and delivered, quarantined or deleted.

4. Design and Policy

As previously indicated, the action taken after identifying an email borne pest depends on

the policies in place. The basis of all policy decisions is that no email message should disappear into the ether; a message should either be rejected during the SMTP session, bounced (returned to the sender), quarantined or delivered to the recipient. A number of different policies were applied, depending on the type of pest and when and how it was identified.

4.1 Virii

All virii should be accepted by the mail system and silently quarantined. No notification should be sent to the sender nor the intended recipient, however the system administrator should be notified via email. Given that nearly all viruses transmitted via email use fake sender address, returning a notification to the person whom appears to be the originator will often result in emailing an innocent bystander. This typically results in additional email traffic due to bounce messages (where the email account does not even exist) or an annoyed person who is getting notifications regarding viruses that they never transmitted, causing them to have mail in their mailbox that they have to delete. Likewise, the intended recipient should not be notified as they do not need to know that someone sent them a virus; it is just more unwanted noise.

4.2 Spam

The policies for spam are somewhat more complex than those for virii, primarily due to the fact that there are multiple points at which it can be identified and the accuracy is not always absolute.

1. An SMTP connection that originates from a host on a designated blacklist will be terminated with a hard error code, causing the message to be returned to the sender.

2. An SMTP session that receives a *MAIL* command specifying a sender address or domain that is on the sender blacklist will be terminated with a hard error code, causing the message to be returned to the sender.

3. Spam that is identified via content analysis will be tagged via the addition of *X-Spam-Status:* and *X-Spam-Level:* headers to the email message, prior to final delivery. This allows for optional filtering by the final recipient.

It is worth noting that this policy was altered slightly when greylisting was introduced, as detailed in section 6.

5. Tools — Selection and Implementation

A number of tools have been used in order to successfully implement this project. This section details the criteria used to select tools, along with the implementation and configuration used to deploy the project.

5.1 Selection

The following criteria has provided a basis for selection. Tools must be:

- Open source
- Secure
- Reliable
- Modular
- Easily configurable
- Low maintenance
- Effective

The tools selected, using this criteria are:

- Qmail[13]
- Ucspi-tcp[14]
- Spamhaus SBL/XBL[10]
- Spam URI Realtime Blocklist (SURBL)[12]
- Qmail-Scanner[16]
- Clam Anti-Virus[1]
- SpamAssassin[7]
- SpamCOP-URI[8]
- OpenBSD[3]

5.1.1 Qmail

The Qmail MTA was selected due to its modular design, along with its high level of security and reliability. Additionally, staff are already familiar with the administration and configuration of Qmail as it has already been heavily used in the past, removing an unnecessary learning curve. Likewise, the ucspi-tcp tools are very configurable and modular, having also been used for numerous years, alongside Qmail.

5.1.2 Qmail-Scanner

Qmail-Scanner is a content scanner and appears to be the obvious choice for performing content analysis on a Qmail based mail server. Written in Perl, it is highly extensible and easily configurable, working with a very large range of anti-virus packages, both commercial and open source. Qmail-Scanner will, depending on configuration, check to ensure that a message is RFC2822[19] compliant, check that MIME types match file extensions, block delivery of

particular file types (eg. VBS and EXE attachments), scan the contents for virii, perform spam identification and more. Many spam messages have headers or MIME boundaries that are not compliant with RFC2822, causing them to be quarantined.

Obviously the overhead of a Perl based scanner needs to be taken into account, given that it is invoked for each and every incoming mail message. In this day and age, computing power and main memory are both cheap, making this far less of a concern.

5.1.3 Clam Anti-virus

Clam Anti-virus is an open source anti-virus toolkit, providing a command line virus scanner, multi-threaded scanning daemon, well maintained virus definition database and database updater. Clam is designed with a number of features specifically for email scanning, including the ability to scan raw mail messages and decompress files that have been compressed using RAR, Zip, Gzip and Bzip2. Both the daemon (*clamd*) and update tool (*freshclam*) are highly configurable and whilst the software is still maturing, the current release (0.72) appears to be highly stable. The virus definition database for Clam Anti-virus is maintained by volunteers and is kept up to date, with many virus samples being provided to the project by users of the toolkit. From experience, it is a very cost effective solution to a commercial anti-virus package.

5.1.4 SpamAssassin

SpamAssassin is a content analyser that uses a wide range of heuristic tests in an attempt to identify email as being either spam or not-spam (commonly referred to as ham.) It is highly configurable and additional site specific rules can be easily added to increase accuracy or to alter behavior. As it is based on a ranking system, each test that is positive will increase or decrease the score assigned to an email message. SpamCop-URI, one of the numerous plugins available for SpamAssassin, is also used in order to provide scoring of URLs appearing within email messages. Its configuration and use is discussed in later parts of this paper.

5.1.5 External Mail Exchanger

In order to resolve problems associated with using the existing external mail exchanger, as provided by one of our upstream ISPs, a third party mail exchanger that subscribes to a number of the same blacklists has been employed as a replacement.

5.1.6 Blacklists

When selecting external blacklists, the requirements change somewhat, as the use of an external RBL has the ability to heavily affect the rejection of email. In most cases they are updated automatically, meaning that use of an untrustworthy RBL could bring an entire email system to a grinding halt. Spamhaus is a very effectively maintained RBL and most entries are a result of significant research. Details regarding IP addresses listed within the RBL are made readily available via their website, including information regarding the reason that it was added to the blacklist. Additionally, Spamhaus is managed responsibly and a high level of reliability is maintained via inbuilt redundancy.

Other RBLs currently available include SPEWS[11], SORBS[6] and ORDB[5], along with the Okean lists[2] that contain IP blocks from China and Korea - countries that appear to be both soft on spammers and have hundreds of insecure and virus infected systems. At this stage, the use of the Spamhaus SBL and XBL lists appear to be sufficient for our purposes and it should be noted that any blacklist in use can have far reaching affects on the ability to receive email, an affect that is passed on to our clients. Any decision to implement additional lists will not made lightly.

The lists maintained by SURBL also use reputable sources, namely Spamcop.net[9], which has earned a fairly solid reputation over time. The use of the SURBL lists is less critical as it will not prevent the receipt of email in this configuration, it can only assist in identifying a message as spam or ham.

5.2 Implementation and Configuration

As identified before, the first line of defense is during the initial connection to the SMTP server. The *tcpserver* program from the ucspi-tcp suite of tools is used to listen for incoming TCP connections on the SMTP port (25/TCP.) When an incoming connection is received, *tcpserver* checks to ensure that the remote IP address is not denied, before setting up a number of environment variables and running *rblsmtpd*, also from the same suite of tools.

If the originating IP address has been manually blacklisted, *tcpserver* will have set the *RBLSMTP* environment variable to a non-blank value, corresponding with the error type and message that should be returned to the SMTP client. Depending on its configuration, *rblsmtpd* will look up any number of DNS based blacklists, in this case the Spamhaus SBL/XBL combined list, *sbl-xbl.spamhaus.org*, is used. If the

remote IP address is listed in any of the blacklists, the SMTP session will be terminated, with either a soft (451) or hard (553) error code, depending on configuration. If the remote IP address is not blacklisted, either manually or within Spamhaus, *rblsmtpd* starts the real SMTP server, *qmail-smtpd*.

Once *qmail-smtpd* has been started, the SMTP session progresses through the normal sequence of events, typically being *EHLO*, *MAIL*, *RCPT*, *DATA* and *QUIT*. If the *MAIL* command specifies an email address or domain that is listed on the local blacklist, upon receiving a *RCPT* command, Qmail will reply with the following:

553 sorry, your envelope sender is in my badmailfrom list (#5.7.1)

This prevents receipt of the message, effectively ending the SMTP session, unless a different envelope sender is provided via another *MAIL* command.

If a valid envelope sender is provided and the SMTP session completes successfully, the message will be queued for local or remote delivery. Instead of invoking the standard queuing component of Qmail, *qmail-queue*, the Perl based *qmail-scanner.pl* script is invoked, via the *QMAILQUEUE* patch[15] for Qmail. In this case Qmail-Scanner has been configured to use Clam AntiVirus, for identification of viruses that are being transported via email messages. If Clam identifies the message as being infected, the entirety of the raw mail message is saved within the quarantine directory and a notification email is sent to the system administrator. Normal delivery of the message is effectively aborted and the message will not be queued for delivery to the intended recipient.

If the message is not identified as a virus, SpamAssassin is used in order to perform spam identification. If SpamAssassin identifies the message as being spam, Qmail-Scanner adds the *X-Spam-Status:* and *X-Spam-Level:* headers, using the rankings returned by SpamAssassin. An additional *Received:* header is added detailing the envelope sender, the results from Clam AntiVirus and SpamAssassin, along with the time taken to process the message. Finally, the message is passed to *qmail-queue* which queues the message for local or remote delivery.

Some of the fine tuning of SpamAssassin should be mentioned, particularly the use of Spam URI Realtime Blacklist (SURBL) and SpamCop-URI. As previously mentioned, many spam email messages contain random words, blocks of text and/or munged words, attempting to thwart spam filters. The URL (or URI) for the website that is being advertised cannot be altered, otherwise recipients of the spam cannot locate the products or services being advertised, rendering the spam useless. Unlike normal IP based realtime blacklists, SURBL contains a list of URIs, or more accurately, a list of domains, that have appeared within spam messages. A number of different lists are made available via DNS queries, although the URI list provided by SpamCop[9] is the most prominent.

Both the *sc.surbl.org* and *ws.surbl.org* lists have proven to be very effective, although a new domain will not be detected until it is added to the list, typically via the submission of spam to SpamCop. Repeat offenders will however be typically caught and the messages tagged appropriately.

A flowchart detailing the implementation and configuration is provided in figure 1.

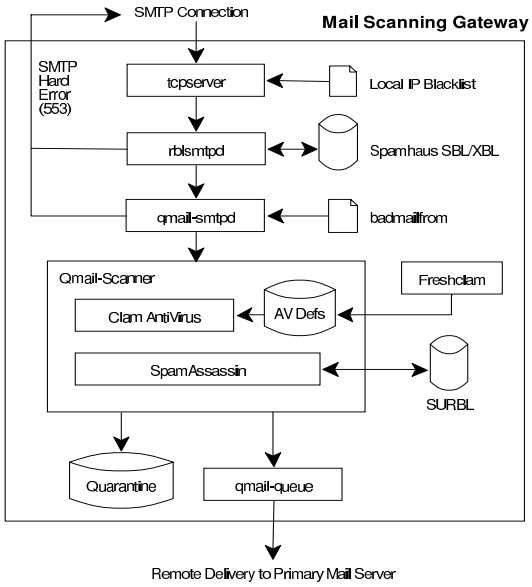

Figure 1: Mail Scanning Gateway

5.3 Deployment and Maintenance

Rather than trying to add the filtering and blocking system onto the primary production mail server, the system was built on a separate server that would scan mail before passing it through to the primary mail server, if it was not quarantined. This allowed for the system to be built and tested prior to being moved into production use, eliminating potential problems caused by the sudden upgrade or changeover. Additionally, the filtering system can be removed from the loop, should the need ever arise, by simply redirecting delivery to the primary mail server.

After the above mentioned software was installed and configured, manual testing was performed, before altering the mail exchanger (MX) records for our corporate domains to use the mail scanning server. When we were satisfied that the system was working reliably,

our clients were notified of the impending changes and all domains were shifted across.

Most of the software used requires very minimal maintenance. The only exception to the rule being the virus definition database for Clam AntiVirus, which is automatically updated on the hour via a *cron* job. Software upgrades occur occasionally, however as with most Unix based software, the compilation and installation is typically seamless.

6. Enter Greylisting

Whilst the use of the Spamhaus SBL and XBL had a significant impact on the amount of spam delivered to our clients, it was still far from being an ideal solution. The primary cause of this is the frequent changes in the systems used to deliver spam, particularly when systems infected with viruses such as Sobig and LoveGate are being used as relays. Such infected systems are a haven to spammers, as it is very difficult to trace the original sender and the number of infected systems allows them to rapidly switch, making it harder to block via IP address. In order to increase our fight against spam, greylisting[17] was added to the above configuration, around a month later.

OpenBSD[3] version 3.3 onwards includes a spam deferral daemon called *spamd*[1][4], that is designed to reject unwanted email in a manner that is efficient to the user and very inefficient to the sender. SMTP connections from blacklisted listed IP addresses are stuttered and delayed by reducing the TCP receive window size (to 1 byte by default) and delaying for a given period of time (1 second by default) between the transmission of characters. This effectively causes the blacklisted sender to send one character of data per TCP packet, wasting bandwidth and time whilst communicating with the tarpit. Put simply, it is designed to waste spammer's resources, keeping them on the hook for as long as possible - over 900 seconds in some cases.

The current version of *spamd* has support for SMTP greylisting, a technique used to filter incoming email based on the behavior of the sender. When an unknown SMTP server attempts to deliver an email message, a soft SMTP error code (450) is returned and the connection terminated, after the *RCPT* command is received. At this point, the tuple consisting of the remote IP address, envelope from address and envelope to address are recorded in a local database. If the same server reattempts delivery of the same message (with a matching tuple) after a specified period of time (30 minutes by default) the IP address is added to the whitelist, allowing future transactions to bypass *spamd* and communicate with the real SMTP server.

A well designed SMTP server will typically retry delivery at regular and decreasing intervals, effectively being the basis on which greylisting is designed. Depending on configuration, delivery attempts will typically continue for up to a week, if a successful delivery has not been achieved. The implementation and use of greylisting prevents the receipt of email from an unknown SMTP server that does not retry at respectable intervals.

Many spammers use custom built applications that attempt to deliver the email directly to the destination mail server and upon receiving a soft SMTP error they will either retry rapidly for a short period of time or give up and move on. This means that many spam senders will never become whitelisted and email will never be accepted from them. A pleasant side-effect noted when *spamd* was deployed, was the reduction in virii being identified and quarantined. It is believed that this is due to the internal SMTP engines in many current viruses behaving similar to spamming applications, whereby retry intervals are not being adhered to.

The flowchart shown in figure 2 details the greylisting service provided by *spamd*, effectively being installed prior to the mail scanning gateway.

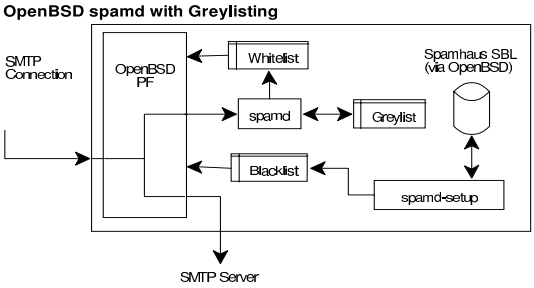

Figure 2: OpenBSD spamd with Greylisting

7. Results

Initially an average of approximately 40,000 messages were being delivered per month, with a large percentage believed to be spam. During the first month of the system being deployed, approximately 2000 virii were detected and quarantined and a further ~8000 messages were blocked due to originating from IP addresses listed in the Spamhaus SBL or XBL. From anecdotal evidence, a number of clients who had been receiving in excess of 100 spam messages per day were now receiving around 5-10 per day.

1. Note that this is completely different to the spamd daemon included within SpamAssassin.

With the addition of greylisting around a month later, the average number of email messages being delivered for the following month was reduced to approximately 22,000, with just 447 virii being quarantined during the month. Once again, anecdotal evidence showed that clients originally receiving in excess of 100 spam messages per day were receiving just 1 or 2 per day.

The primary result is a lot of happy customers who no longer have to delete hundreds of spam and virii from their inbox, saving both time and bandwidth for all concerned.

8. Future Work

A number of areas exist whereby the system could be further enhanced or extended. While a lot of data is generated by the various components of the system, at this stage none of it is collated or graphed, requiring the system administrator to watch numerous log files. Closer monitoring and reporting would be beneficial as this would reduce the amount of ongoing work required, as would it increase the feedback regarding the effectiveness of the system.

OpenBSD's *spamd* lacks the support for a number of commands and as a result, does not appear to be fully compliant with RFC2821[18]. This has caused at least one problem due to a strangely implemented mail relay issuing a *RSET* command prior to attempting delivery of email. The *RSET* command is currently unrecognised by *spamd*, resulting in a hard error code and the email being bounced. This command could easily be added, along with a number of other commands required by RFC2821.

Given the recent release of Cabir[20], the first proof of concept worm written to target mobile phones, the author would be interested in exploring the application of filtering to mobile and ad-hoc networks. Whilst Cabir is not considered a real threat, due to numerous technical complexities, it is quite feasible that further mobile phone viruses will be written in the future, ones that may spread with greater ease and be more destructive. Additionally, there are problems with SMS based spam, analogous to email based spam although transmitted via SMS messages, although this appears to be more of an issue overseas than in Australia.

9. Conclusion

As shown in this paper, email borne pests, can be effectively combatted using a carefully designed mail filtering system. The implementation of this project has only used open source tools and technologies, creating a cost effective, secure and reliable system; a system that can easily be replicated and deployed elsewhere. The more mail servers that deploy techniques to combat these pests, the harder it will be for spammers to effectively deliver their wares.

There does not appear to be a single solution to email borne pests, rather a multi-layered approach is required in order to identify and eliminate as many as is feasibly possible. Whilst it is always possible for some pests to bypass the mail filtering system, a significant reduction is still far better than none, increasing the usability of what is arguably the world's most important communications media.

Bibliography

[1] Clam antivirus, http://clamav.sf.net/.

[2] Okean Asian Spam Blocks, http://www.okean.com/asianspamblocks.html.

[3] OpenBSD, http://www.openbsd.org/.

[4] OpenBSD spamd, http://www.openbsd.org/spamd/.

[5] ORDB - Open Relay Database, http://www.ordb.org/.

[6] SORBS - Spam and Open Relay Blocking System, http://www.sorbs.net/.

[7] Spamassassin, http://www.spamassassin.org/.

[8] Spamcop-uri, http://sf.net/projects/spamcopuri/.

[9] Spamcop.net, http://www.spamcop.net/.

[10] The Spamhaus Project, http://www.spamhaus.org/.

[11] SPEWS - Spam Prevention Early Warning System, http://www.spews.org/.

[12] SURBL - spam URI Realtime Blocklist, http://www.surbl.org/.

[13] D. Berstein. Qmail: the Internet's MTA of choice, http://cr.yp.to/qmail.html.

[14] D. Berstein. ucspi-tcp, http://cr.yp.to/ucspi-tcp.html.

[15] B. Guenter. QMAILQUEUE patch, http://www.qmail.org/qmailqueue-patch.

[16] J. Haar. Qmail-scanner, http://qmail-scanner.sf.net/.

[17] E. Harris. The Next Step in the Spam Control War: Greylisting, http://projects.puremagic.com/greylisting/whitepaper.html, 2003.

[18] J. Klensin. Simple mail transfer protocol. RFC 2821, 2001.

[19] P. Resnick. Internet message format. RFC 2822, 2001.

[20] Symantec Security Response. Symbos.cabir, http://securityresponse.symantec.com/avcenter/venc/data/epoc.cabir.html.