

STOPPING MALWARE AT THE GATEWAY – CHALLENGES AND SOLUTIONS

Martin Stecher

Secure Computing Corporation, Vattmannstr. 3,
33100 Paderborn, Germany

Tel +49 5251 8717 525

Email martin_stecher@securecomputing.com

ABSTRACT

Anti-malware scanning at the gateway differs with regard to its requirements from anti-malware scanning at the client or server.

Some aspects become simpler, for example, there is no on-access scanning, and false positives are less dramatic, while new challenges are introduced, such as performance and update issues, latency, chunk-by-chunk scanning, download progress indication and support for archives and alternative document formats.

Which protocols should be handled by a gateway? Is SSL scanning possible and needed? Can callout protocols, such as ICAP or OCP, help to write an application-agnostic scanner that works in all environments?

MOVING TO THE GATEWAY

While anti-malware solutions on clients and servers are still essential, there are many reasons why additional protection at the network perimeter is desired by security-sensitive users: rootkits and malware downloading via connections that may have bypassed the desktop firewall, such as BITS (*Microsoft's* Background Intelligent Transfer Service), bring the possibility that malware might circumvent client or server software; if malware makes it on to a client, it often tries to shut down anti-malware software first, preventing further countermeasures; a portable computer plugged into the corporate network may not have up-to-date protection.

So, now that gateway solutions are becoming more and more popular, what is needed to get to such a product? Can I just use any on-demand client scanner and attach it to an HTTP proxy or SMTP gateway? In fact, most gateway products today use the same anti-virus engine in their core as in the client product, and a good and solid anti-virus core scan engine is certainly application-agnostic. The status of an early prototype can quickly be reached with this approach, but there are still many challenges and pitfalls. Some of them, together with potential solutions, will be discussed in this paper.

SUPPORTED PROTOCOLS

A gateway solution is able to see data packets, but most malware can only be detected if the payload, which is split into many packets, is reassembled. This can only be achieved if the solution understands the protocol, usually through being a proxy server for those protocols.

The primary protocols that first come to mind are: HTTP and FTP for web and SMTP for email. This might be

supplemented by some Instant Messaging (IM) and P2P protocols or other email protocols such as POP3 or IMAP, in case the particular email server is not deployed in the intranet, but must be accessed through the gateway. Most other protocols can either be forbidden completely, should carry no real payload (DNS, ICMP), or are used for trusted site-to-site traffic (VPNs, SSH).

But what about HTTPS? Downloading infected attachments from web mail via HTTPS is something that can happen easily, and it seems to be just a matter of time until a downloader simply switches to HTTPS to download additional files. Scanning within these encrypted connections is a challenge for the gateway solution; without knowing the session key, it is impossible to break into the secure end-to-end connection (that's why we use HTTPS). The gateway has to implement a trusted intermediate for HTTPS, which means it must maintain separate HTTPS connections to client and server, and deal with certificate verification policies, creation and signing of new certificates and secure storage of private keys. While this is relatively simple for a reverse proxy deployment (an anti-malware gateway in front of a web server), where only a known set of certificates needs to be handled and can be shared, it is much more complex for a forward proxy scenario (a gateway that protects an intranet); the number of HTTPS scanner solutions for that purpose is still very low today.

SUPPORTED DATA FORMATS

A client-based solution has an on-access scanner which solves most issues of unsupported archive formats. Whatever exotic archive format is being used, the on-access scanner gets a chance to detect the malware when the user's archive tool expands the archive.

The gateway solution cannot know what a user will do with the data. Each file that may contain objects either needs to be scanned or it should be denied access completely. Allowing those formats to pass without scanning is a security issue and should only be done as an exception on the whitelist. The list of formats not only includes all kinds of archive file types, but also alternative data formats for *Office* documents with potentially infected embedded objects, such as *Office* WordML and RTF documents.

The on-demand scanner part of a client solution usually scans archives to a certain depth of nested levels and then stops scanning (eventually reporting an error), and also stops scanning if an archive or *Office* document cannot be decrypted.

With the absence of on-access scanning, the default setting for a gateway solution should be different. Archives with dozens of nested levels are usually not business-relevant and blocking those by default is acceptable; the same holds true for documents that cannot be extracted because they look corrupted (maybe they are corrupted, but often enough there is a client tool that is still able to extract that corrupted file).

PERFORMANCE

A client-based anti-malware solution easily scans on access all data that is handled by the user and usually does not take more than a few per cent of the CPU time. A gateway solution needs to do the scanning job for hundreds, thousands or even hundreds of thousands of users. What if these users access 1,000 URLs per second and receive 25 email messages each

and every second? Will the solution scale to those numbers and beyond? Can multiple machines be deployed in a cluster and will they work together? A gateway solution usually does more than pure anti-malware scanning, but also adds user authentication, caching, message queuing, reporting, etc. A cluster deployment is usually much more complex than a single-box deployment. However, the anti-malware scanning is usually the part of a gateway solution that takes most of the system resources, especially many CPU cycles. This means that the performance of a complete gateway solution scales almost linearly with the performance of the anti-malware scanner.

Obviously, the performance of the overall system increases significantly if the scanner is bypassed for file types and conditions that are known to be free of malware. But who would claim to 'know' that a file type is really malware free? At least we can diminish the bypass condition to 'file types for which it is known that the malware scanner won't find anything', as this will not have more false negatives than the 'scan all files' approach.

There may be certain file types (or media types or content types) for which no signature or rule exists in the malware scanner and which can therefore be used for bypassing. A common pitfall in that approach is that the file type is not validated before bypassing. If the solution only checks for the file name extension (in the URL or email attachment), it is very easy to fake the content type. Also just checking for the Content-Type header of the application protocol is not much better, as this information is usually provided automatically by the web server, determined again solely by the file name extension.

But some client software does not rely on this provided information, and instead tries to determine the media type of the data itself. Try, for example, the following: create a simple HTML file, have it start with the '<html>' tag and put it onto a standard web server, but name it 'test.gif' instead of 'test.html'. Then use *Internet Explorer* to request this file. The web server should provide the file via HTTP with the 'Content-Type: image/gif' header (because it only sees the file name extension); *Internet Explorer* will instead check the file content, recognize that it is not an image, but an HTML file and deliver it; *Firefox* displays a broken image symbol. (A plain text example with the Eicar test file sent as GIF can be downloaded at the csm-testcenter [1]).

If the gateway solution only checked the file name extension or Content-Type and decided to bypass GIF images, it would not be able to detect a malicious script embedded in that HTML file. This means that the gateway solution must be at least as good as typical client software in detecting media types of files if it wants to let certain files bypass the scanner. Content type analysis is a must, and if in doubt, don't let anything bypass. Also, the solution should keep in mind that information about media types for which no signature exists is not persistent. Remember the time when vulnerabilities of JPEG imaging tools had been disclosed and people expected anti-malware tools to be able to block these exploiting images: bad, if a gateway solution had been using JPEG on a fixed bypass list at that time.

LATENCY

While this is not an important criterion for email, it can be very important for web traffic. It goes hand in hand with

performance, and yet is different. For some file types, such as HTML files, a browser can start rendering even if only the first part of the file has been received. Many anti-malware gateway solutions increase the latency significantly because they wait until the complete file has been received from the Internet at the gateway before they forward any data to the client. The reason is that the typical on-demand anti-malware scanner needs to see the complete file before it can do its job. This is probably the parameter that reveals most the client-based origin of most engines. For some media types, chunk-wise scanning and forwarding of data will accommodate the browser's capabilities and maintain the end-user surfing experience from direct Internet connections.

For most large downloads (for example, an archive with some files), the browser is not able to do anything with the data before the complete file has been received. One might think that latency is not an issue in this case, but in fact, download progress indication is a very important instrument for end-users. If you click on a download link and just see that the browser is doing something, but get no further indication as to how long it will take and how much data has already been received, you are totally lost as an end-user, with no clue whether the download is hanging, whether you should retry or whether it's time for a coffee break.

In particular, the download of an archive can be a lengthy process at the gateway: it might be a very large file that has to be retrieved from the Internet first and then extracted and scanned. Not forwarding any data to the client during that time could cause a connection timeout at the browser, and at the very least it will make the end-user unhappy.

Multiple strategies have been invented to work around this problem:

Data trickling

After receiving a certain amount of data, the gateway forwards a small part of the data to the client. This is the simplest method and prevents a client from timing out; it also indicates to the end-user that there is some progress, although the estimated time calculated by the browser is then much too long. The question is how the data trickling should continue after the complete file has been received at the gateway, while the product is still busy with lengthy archive extractions and scanning.

Further problems occur if the infected piece is at the beginning of the file and may have been sent to the client already in data trickle mode, or if an error message can no longer be displayed to the end-user after the first part of the original file has already been sent; the end-user will then only see that the download has been cancelled, but does not know that this happened due to a detected data infection.

Patient pages

While the originally requested file is being downloaded to the gateway, the proxy entertains the end-user with intermediate HTML pages that show the progress of the download. This solution usually looks best to the end-user, but also has issues: an end-user right-clicks to 'Save target as...' and all kinds of download tools are likely to store the intermediate page rather than the real file, and due to security awareness, browser vendors implement stricter methods with each release, which can cause some patient page techniques to fail.

Separate queries

By installing a separate tool, such as a browser plug-in, additional queries can be submitted to the gateway to ask for the progress of an ongoing download. Beside the issue that this plug-in needs to be rolled out to every client in the network, this method requires that the gateway is able to respond to the query on a separate connection; especially in a cluster this requires an additional synchronization effort.

Late clearance content encoding

In 2002, I published an Internet draft [2] about a new way of content encoding that would solve the problem without the disadvantages of the methods mentioned above. If supported by a browser, HTTP defines standard headers to indicate the support and usage of that encoding. A file that is downloaded by the gateway is encrypted with the AES algorithm chunk by chunk, as it is received and forwarded to the client. After the complete file has been received (and forwarded in encrypted form), the decryption key is sent at the end only if the anti-malware check was negative, otherwise an end-user error message is sent. The browser knows how to handle all different use cases, and tools not supporting this method will not get any data with this way of encoding.

Although several reviewers and also some developers from a large, commercial browser company responded very positively, this method has not been implemented so far in any browser.

UPDATE STRATEGY

Every anti-malware solution needs updates of signatures, lists, rules or whatever, and updates of the engine. For most products, the update frequency has increased a lot during the last few years. While for client- and also server-based products, a short lock-out period during switching from the old to the new instance is not an issue, it can very well be an issue in a gateway deployment if thousands of requests have to be handled every second. A decent gateway product should be able to update to new versions of its anti-malware engine without any performance degrading. For that purpose, existing requests should be handled by the previous instance and new requests by the new instance, so that a handover is not performed until all existing requests on the previous version have been fulfilled and that instance can be unloaded.

FALSE POSITIVES

False positives of a client or server installation can be dramatic; such a product should not quarantine an elementary system component out of the system folder, nor must it delete the only copy of the user's *Word* document containing the dissertation he or she spent the last year working on.

It is very justifiable that tests of anti-malware solutions attach great importance to a zero false positive rate for such products.

The matter looks a bit different for gateway products, though. A false positive remains a nasty thing, but usually is not that dramatic. A system should always continue to work, even if a download of a file has been blocked (or failed for other reasons). Blocking files that the user wants can be fixed or worked around (all programs should have a whitelist function) and the data retrieval can be repeated (new

download from the Internet, ask the sender of an email to resend, etc.). By nature, a file retrieved through a gateway is not the only copy and still exists somewhere else.

A false positive on a web page might, furthermore, not necessarily mean that access to the page was blocked, but rather that scripts on the page have been removed. So, in spite of the false positive, the page remains basically accessible.

In the section about archives, I listed some arguments as to why a gateway solution should block a file when in doubt; this always goes together with a number of false positives.

The more relaxed attitude with regards to false positives also opens a number of opportunities to increase the detection rate, especially of new and unknown threats. Methods for proactive security almost always come at the price of a higher false positive rate, but as gateway protection is especially for those users that want a stricter policy against attacks from the Internet, the gateway seems to be the ideal place to deploy new and advanced proactive security methods.

ICAP AND OPES – CALLOUT SERVER DEPLOYMENT

So far, gateway anti-malware solutions have been described as sitting on an application layer proxy at the network perimeter. Often, though, proxies may have been deployed already, and not in all cases can the anti-malware gateway be a full replacement for an existing proxy. The anti-malware gateway can then be installed in-line with the existing solution, which increases the proxy chain at the gateway. This complicates the deployment and leads to errors: how should the solutions be aligned in the chain? Which proxy needs to do proxy authentication for the end-users, and can it access that data from its place in the chain? How should load balancing and fail-over be implemented between the elements in the chain? How well is each product in the chain prepared for unexpected server responses from the Internet? How well do the proxies interoperate?

In 1999, some companies started to look into placing gateway filtering solutions beside the normal proxy chain architecture. The idea was to vector out the data from the proxy server to one or many callout servers that would then filter the data and give the results back to the proxy. Some first pre-versions of a new protocol had been developed, and in 2001, version 1.0 of the Internet Content Adaptation Protocol (ICAP) was complete. It took some more time to discuss this protocol in several standardization bodies (ECMA and IETF) and since 2003 ICAP/1.0 has been available as Informational RFC 3507. In those years, it became the *de facto* standard for callout protocols and many companies have implemented it in their proxy servers or filtering products. Today dozens of companies are supporting the ICAP idea as participants in the ICAP forum [3].

ICAP has been exclusively designed for handling HTTP traffic (but has also been used for other similar protocols since). The proxy server acts as an ICAP client, encapsulates HTTP requests and/or responses into ICAP messages and sends them to the ICAP server, which might be an anti-malware gateway product. This ICAP server can then return the data as is (or indicate that it did not modify the data), it can replace the data with an end-user error message, or it can modify the content to remove infected parts of the file. The proxy server remains solely responsible for proper

HTTP handling between client and server, it does the proxy authentication and is able to forward user credentials and also group information that has been verified already to the ICAP server as standardized meta-header information.

There are two central features of ICAP that help to increase performance and throughput and to decrease latency:

- Preview: the ICAP server can send only the first part of a file and then waits for the ICAP server to provide feedback as to whether it is also interested in the rest. If not, the remainder of the file will be sent directly to the client, without vectoring it out to the ICAP server. This method can be used if the anti-malware engine is able to determine after inspecting the first part of the file that it is not infected.
- 204 response: this is the response type sent back after the preview if the server is not interested in the rest of the file, but it can also be used at the very end of the file (if supported by the ICAP client). This means that the file does not need to be sent back and the proxy can just use the original data, so it saves half of the bandwidth. Unfortunately, it also means that this feature increases the latency a lot.

Just while ICAP was developed, a number of interested people wanted to set up a group within IETF that dealt with callout protocols in a more general way. Discussions took too long deciding whether this group could really become a working group within IETF, but finally in February 2002, the OPES WG (Open Pluggable Edge Services Working Group) was approved [4]. Unfortunately, the Internet bubble had burst in the meantime and the number of contributors to the working group had dwindled since the official founding.

OPES analysed ICAP and listed the pros and cons that came up over time and then concluded that a potential successor of ICAP/1.0 should be a protocol with some major changes instead of just some minor tweaks. The result is a protocol with the working title OCP (OPES Callout Protocol), but which could also be renamed ICAP/2.0. Unfortunately, no commercial product has implemented OCP so far; ICAP/1.0 is still working too well and there is too little pressure to upgrade to the new version. Upgrading from existing ICAP/1.0 implementations would also cost some development resources, as the technical protocol differences are significant. And due to a lack of interest and participants in OPES, the working group wound up in March this year after working through its first two charters.

Nevertheless, OCP has some interesting advantages over ICAP/1.0:

- The protocol core (RFC 4037) is application-agnostic, which means that in contrast to ICAP it has been designed to be used for all kind of protocols. The OCP agents negotiate a profile that should be used for an application transaction. An HTTP profile has been developed and standardized as RFC 4236 and an SMTP profile has been prepared.
- The OCP clients and agents can send multiple transactions on a single transaction in parallel, and sending and receiving is fully asynchronous. There is no wait-for-an-answer condition by design, as in ICAP, for the preview response.
- Multi-stage ‘previews’ can be used. In fact, the OCP filtering server can request at any time during the

transaction handling to get out of the loop because it finished its task earlier during the transaction.

- OCP client and server can dynamically negotiate about the data that is preserved on the proxy side and that the filtering server can refer to without sending it back. The OCP client can use whatever memory it has available, but is not obliged to store a complete transaction if it is running out of resources and the OCP server can also just refer to parts of the file and add other modified parts in its response.

OUTBOUND PROTECTION

The focus of an anti-malware gateway is primarily inbound (where ‘inbound’ means into the guarded network and can also be a server if the gateway is a reverse proxy). But the gateway is also a very good deployment option to detect whether a device within the guarded network has been infected. If a detection method is added for a new piece of malware, a client-based solution may not be sure whether an infection that occurred before the update can really be detected or whether the infected files are effectively hidden and cannot be seen even by the updated client anti-malware tool. Likewise, a mobile worker may plug his spyware-infected laptop into the corporate network – infected, because he or she didn’t bother to update his/her client AV’s expired licence, for example.

But worms, trojans, and spyware want to exchange data with the Internet, and this outbound traffic will also pass through the gateway. The gateway anti-malware solution should also be able to scan that traffic, stop this unwanted data transfer and notify administrators and end-users about the infection of the client. In order to inform the end-user, some out-of-band notifications need to be used, either by blocking other normal HTTP traffic (presenting the end-user with an alert notification when surfing the web), or by sending an email notification where the email address of the user of an infected client is known.

GATEWAY SOLUTION TESTING

Most of the anti-malware product tests and comparison tests focus on client and server programs. Sometimes the test methods allow gateway products to participate in the test and sometimes, as with VB100, this is not possible. Some certification tests have been developed especially for gateway products.

The main criterion for whether a test is suitable for gateway products is the on-access scanning part, which does not exist for a gateway. Neither, in fact, does a gateway have an on-demand part, but only an ‘on-data-through’ part – though the difference between the two is not great.

We have also seen that the false positive rate of a gateway product must not be interpreted in the same way as for a client-based solution, although many tests do not pay attention to this.

Also for the detection rate, it is questionable whether all kinds of malware need to be detected by a gateway solution: a piece of malware that is dropped by a downloader into the system folder, but is never transferred as an isolated file over the network, would never pass through a real gateway and may therefore not be the best test pattern in a test lab environment.

On the other hand, performance testing by doing many parallel requests, instead of measuring the time used to go sequentially through a test set file by file, would be a worthwhile addition for a gateway test; the same goes for latency time and would be a useful progress indicator to end-users.

The performance test set should use real files instead of random data and it should also have a file type distribution similar to what a real gateway in a customer deployment sees. Otherwise the performance test cannot measure correctly the optimization strategies discussed earlier in this paper.

REFERENCES

- [1] EICAR test string downloaded with Content-Type: image/gif. <http://www.csm-testcenter.org/cgi-bin/eicar.gif>.
- [2] LateClearance Content Encoding (Internet draft October 2002, expired April 2003, copy at <http://www.martin-stecher.de/draft-stecher-lclr-encoding-00.txt>).
- [3] The ICAP-Forum. <http://www.icap-forum.org/>.
- [4] The OPES WG wound up in March 2007; the mailing list remains open. <http://www.imc.org/ietf-openproxy/mail-archive/>.