Implementing Antivirus Scanning for a iPlanet/SunONE Messaging Server: Decision Methodology and Implementation Examples

Eric Straavaldsen
GSEC Practical Assignment
Version 1.4 Option B

Abstract:

Document a procedure to implement virus scanning on iPlanet/SunONE Messaging server while maintaining availability and performance. Directed not at the step by step of implementing AV scanning but more a procedure to do the decision making process and a look at major options. Also showing an example of the due care needed when implementing new security functions onto an existing system.

Scenario Overview

In the spring of 2002 I was the technical head of a project to add antivirus scanning to a large installed iPlanet/SunONE Massaging Server cluster at a higher education campus. The implementation of an antivirus scanning solution for an iPlanet/SunONE Messaging Server cluster can take one of two basic methods. One solution type was to use a scanning gateway; we categorized this as our non-integrated solution. The other solution type was to integrate a scanning engine directly into the messaging cluster; we categorized this as our integrated solution. I will endeavor to describe how our team examined the options and then proceeded to test and implement our choices in scanning agents to have minimal impact on the performance and availability of our mail system.

Product Selection and Options

When the antivirus team was formed we started contacting vendors in order to build a comprehensive list of the options available to us. Our first task was to form a set of criteria to distinguish between the numerous vendors of antivirus software in the market. Some of the factors we considered were, initial cost, ongoing cost (in both vender fees and our time), vendor certification, comparable experience by vendor (case studies), responsiveness of the vendor, and consideration of what other antivirus vendors were currently deployed at our campus.

One of our basic tools for pairing down the vendors to contact for a solution was their antivirus certification. It was believed that independent certifications would be an acceptable baseline for checking quality of various vendor products. The ICSA Labs Antivirus Certification[i] was the first threshold we chose to look at. Our research into antivirus certification resulted in the discovery of other groups such as, West Coast Lab's Ceckmark[ii]. Which also provides a good tool to check the antivirus selection to see if it passes a minimum standard of quality.

The other criteria that should be described, due to its unusual nature, was why we were taking into consideration what other antivirus products were deployed at our campus. Our team felt that since we knew that there was no one vendor that would always be the first out with a definition for every new virus threat. There was a slight risk in not diversifying our antivirus solutions. By making it a criteria that we had different vendors at least being explored in our mail system solution from our desktop solutions we could get a little more protection from the different approaches the vendors took towards scanning. This was not to rule out what vendors were widely deployed already from being chosen, but rather to promote the idea of layering our security. While not seen as a vital part of the decision making process it made us include several more vendors in our test than we would have otherwise looked at.

With these basic criteria we were able to compile a list of our top vendors. By doing this work we were able to have a level playing field of expectations of our vendors. From this list we were able to start our selection of solutions.

Non-integrated Solution

The first and simplest solution that we examined was that of a non-integrated solution. This we felt would be the simplest route to implement to our initial charge of implementing virus scanning for the mail system. The two major choices in this matter were either a new server running sendmail with a scanner from the milter[iii] filtering system, or an appliance.

The stand-alone server doing the scanning task was quickly ruled out because of the added cost of local system administration time over that of an appliance solution. Our pricing exercise was that buying the new standalone server and software for it was not substantially less expensive than an appliance. So we turned our attention to appliance solutions as the less expensive non-integrated solution option.

The other advantage that an antivirus appliance had was that it required no changes in our existing mail system. Its drop in nature also meant that we could add and remove it from the mail system function without disturbing the configuration in anyway. This simplicity was extremely attractive to us because it would allow us to add this important security function without altering an existing, complex system.

The primary disadvantage was that we could not guarantee that we would scan every message in our system. Messages that were outbound, sent within the system, or from local native clients would not be scanned by the appliances. Local native clients were a sticking point for us. Since the logical placement of the scanners was dependant on DNS MX records, and that most mail clients do not support MX records we could not just set them logically between mail client software and the cluster. Previous experience had also shown that getting 80,000 end users to all change their client settings was unlikely to be practical, or timely. So we proceeded under our initial charge, and from the collected best practices of our vendors this was an acceptable solution, our research also showed this solution considered a best practice by many other organizations.

The initial configuration for the appliance solution was very simple, requiring only three real configuration functions. First to maintain availability we chose to purchase two appliances, each capable of handling about 75% of our peak load. With one we felt able to handle a normal load, so the loss of one appliance would most likely allow us to continue to function. The appliances then were configured to know what mail systems they were scanning for. The last change was in our MX records to point to the new antivirus appliances.

Our configuration had our DNS records set with MX weights of 10 for each of the two antivirus appliances with the iPlanet/SunONE Messaging SMTP gateways to 5. By keeping the SMTP gateways in the MX allowed us to be sure that we would not lose mail in the case of overwhelmed appliances or failure of appliances. As a team we were quite happy to find that "turn key" solutions could in fact be turn key.

The choice to keep the cluster MTAs in the MX cycle was because the principle of availability was more important to our end users at the time than perfect scanning coverage. The occasional virus infected email would be more acceptable than mail being lost or delayed significantly. To accommodate this problem of unwatched infection vectors the team suggested that all desktops and laptops have antivirus software installed as both a protection against over flow or failure of the appliances but to also cover other infection vectors like floppy disks and other mail services.

The use of AV appliances served us as a very effective and easy to implement solution. The modular nature also allowed us to logically place it in front of other mail systems run on campus. The solution worked very well as a solution for scanning the inbound email.

Integrated Solution

The initial implementation of a non-integrated choice gave a good initial solution for our iPlanet/SunONE Messaging server, and for many organizations it may have been a good final solution. Our change to an integrated solution from an non-integrated solution was driven by a request for enhanced functionality. Management felt that by moving our solution into our Messaging cluster it would allow us to provide a more complete scanning protection, not only to our campus but to the internet as a whole. With a new direction and standard set we the chose to pursue the other major solution option, placement of the scanner on the iPlanet/SunONE Messaging server cluster.

Placement of the scanning internal to the iPlanet/SunONE Messaging cluster had several key advantages for us with only a few draw backs. The most significant advantage for us was that we could be sure that we would scan all mail passing through our system, inbound, outbound and internal. It also simplified the scanning of hosted multiple domains on our mails system. The drawbacks were primarily of the difficulty of configuration and possible scaling issues.

Testing

The implementation of the integrated scanning solution was significantly more involved than our appliance scanning solution. Our team started with a testing strategy to both test our install configuration and performance changes to our system. With this plan we endeavored to get a proper grip on this second phase of our project.

To perform our test we need to gather a set of tools to do our test. The four tool components we needed were an eicar test file, mailstone, a script to help watch the mail queues and a set of scripts to test performance of the actual engine. The first tool to acquire was the eicar test file[iv] as a foundation for the later tools. The basic tool for testing how the scanning engine impacted the performance of the iPlanet/SunONE Messaging cluster was mailstone[v].

Our mailstone configuration was setup to only test the SMTP function of the Messaging cluster, for this was the only part of the system that was known to be directly changed with the addition of antivirus scanning to the cluster. The mailstone configuration was relatively simple. We chose to emulate a load that was half virus infected. This was a trivial task to set it up to do this. All that was needed to balance it was to source each of the test messages in the configuration file, and mailstone would generate a load divided into equal shares between the message files in the configuration. We simply used one of the generic test emails that came in the package. We selected one to be for the clean message and modified a copy of the same test message to contain a copy of the eicar test file. With this static test mailstone configuration on an idle server we were able to isolate out the performance differences in our AV engines and SMTP configurations.

While we ran the tests we could monitor system performance with tools like orca,[vi] sar, mpstat and iostat but this did not answer the question of what the scanning agent was doing to the end user experience. Part of the solution to that was to check how the SunONE mail queues were being affected. A baseline was established before the scanning engine was installed. With the “imsimta qm dir –tot” command it was possible to monitor the system’s queues. The other part of the solution was to inject a message into the test or randomly select a message from the load test and examine the headers to see how long delivery was taking. We almost exclusively chose to randomly select a handful of the mailstone messages from the test to check headers for delivery information.

With the tools and baselines in place, we were now able to start a proper evaluation of our implementation. Our initial expectations were based on the iPlanet/SunONE estimates that 30% of our capacity would be lost due to the addition of AV. Our initial test runs of 1,000 messages seemed to hold up this assumption, but subsequent runs of five, ten, and 20 thousand messages seemed to show a substantial growth in our mail delivery queues and in delivery time. When the system queues were only around 100 messages we found that our delivery time was around one to two minuets. Larger queues grew message delay by what seemed to be more than simple linear growth. This rapid growth of mail delay drove the concern to make the scanning as efficient as possible to keep mail queues down and improve how fast we could clear them after a message storm passed.

For us to acquire accurate timings of the script (invoked by the iPlanet/SunONE Messaging server to handle the actual scanning function) and AV engine we wished to use the Unix time command, to show us how much time was being spent in execution. To achieve this, we dissected the script to find what was needed to run it outside of the messaging server environment. With the knowledge gained from that we developed one test script to run the whole scanning system and one to test script for the AV engines. With these two scripts, we were able to see how tuning actions would impact performance. (See the Appendix for example script).

Since the way iPlanet/SunONE does internal virus scanning is by running a script, against the various message segments that the product breaks an email message into, our initial concern was over the startup cost of a shell script. So the first script we wrote was to run just the scanning engine against a file repeatedly (see Appendix) and log the time the scan took. On our test system (a SunBlade 100) the first script produced an average of about 1.15 seconds for the engines to scan a test file.[vii]

The next target for analysis was the iPlanet/SunONE virus scanning script. After much script reading and debugging, we produced a second wrapping script (see Appendix) to test the whole infrastructure for scanning that was going to be run under iPlanet/SunONE. We found that raw startup of the iPlanet/SunONE virus scanning script was not a major part of our processing time for a message. The average time the whole scanning infrastructure took was 1.2 seconds. While the iPlanet/SunONE virus scanning script did add some overhead it was not as large as initial fears.

All of the tested vendors came within a few percentages of each other's performance. So with similar performance from all our vendors we could assure ourselves that there was little we could do in the scanning subsystem to gain much in the way of a safety blanket, in the form of excess capacity. Though we did try both a memory mapped file system for the scanning subsystem and a very optimized shell to run the script, neither of these trials gave any noticeable improvement in performance.

With all of the basic feasibility and performance testing done the, team was ready to propose an integrated solution. In our estimation, the added load of antivirus scanning would drop our excess capacity below our busiest day projections with the current cluster hardware, and that new hardware would be required to support the initiative. A second recommendation was that in the open environment of higher education we could not trust that scanning at the central mail system would be adequate to meet our virus protection needs. So we proposed that the system be augmented with a scanning solution for the desktop workstations on campus. The recommendation was accepted for immediate implementation on the cluster and with desktop protection to form part of a later project.

Implementation

As we went forward in implementation based on the script files and fragments from iPlanet/SunONE[viii] there were several aspects we felt needed to be changed for it to be a supportable system. We needed to: alter the script provided from iPlanet/SunONE for our needs, build an automation script to update our virus data files, and a procedure to update the AV engine and tuning performance.

The scripts provided by iPlanet/SunONE were built as a set of tools to do several functions beyond AV scanning. To keep the task of updating and modifying the scripts as easy as possible we chose to simplify the provided scripts to remove all the unused functions. By this we hoped that we could lower the cost (in both memory and CPU utilization) of running the script (if even by a tiny amount). We also added a feature to find out the name of the virus found and include it in the notification. The latter was done by a simple grep statement added to the ScriptFunctions script in the substitute_part segment. This would capture the virus name from the scan log of the file and add it to the notification message. Also we, chose not to notify the sender, but only pass the notification to the recipient. We chose to first try to clean the virus from the file, but in the case that we could not clean the virus, we chose delete the attachment and add the message about what virus had caused the attachment's deletion.[ix]

The choice to not notify the message sender of the infection was of great importance when the Sobig.f virus came out, several of our campus users found their email unusable because of the large numbers of notifications that were sent to them as a result of the Sobig.f virus sending large numbers of messages purported to be from them. This choice was of some contention in our group, but we eventually felt that we were better not sending, than ending up in a situation where our virus scanners were causing a DOS attack on a remote site.

We examined the script provided by our engine provider (NAI). The script found in the documentation was not satisfactory. It did not have what we felt was adequate level of error checking. After searching for other work written to do NAI dat file updates, we found the script provided to update the NAI dat files from the FreeBSD project. This gave us a good basis for what we needed to build a script that met our standards (example of our script can be found in the Appendix). The primary alteration beyond what was needed to run it on Solaris was, that there must be a check to make sure that the scanner did in fact work after an update. This addition was vital for us to be sure that we would know that after an update we would have a functioning scanner. We also tested the impact of scanning mail when the script was running to see if we would miss viruses. After a series of tests we felt that the risk of missing a virus (or damaging/losing an email) while running this script was low enough that we could use the script whenever there was a new virus threat.

The AV engine updates turned out to be an invasive procedure. It was fairly clear that the engine could not be updated on an active system. So to ensure availability we had to monitor our mail load and select a time where we could take one of our SMTP hosts offline when it would not impact our mail delivery. The order of action was quite simple, we first stopped the tcp_smtp_server service that handles inbound traffic and then used the queue watching commands that are part of the iPlanet/SunONE server to find out when the conversion channel (the section of the system that the AV scanner runs under) was clear of messages. Then run the NAI installer to do the update followed by the dat file update script. After we knew that the dat files were current we would start up the tcp_smtp_server subsystem and watch the conversion and virus logs to see if we were catching viruses before we could call the update complete.

The tuning of virus scanning on iPlanet/SunONE is more a matter of trial and error for the specific hardware running it. The two relevant files are the job_controller.cnf and the imta.cnf. It is in these two files that you set the number of jobs available to run the conversion channel (the channel that runs the AV scanning engine). The selection of how many jobs to have available for scanning is a balancing act between building up mail queues and over loading the systems. While it is not simple to give a formula of how to set these for the specific system this is running on, an example of what we have found a good balance for a Sun 280R running only as a SMTP server can be found in the Appendix, and is probably a good starting point for tuning the system. While tuning, the most important things to watch are system load[x] and queue depth (a good reason to have done some initial performance profiling). If either of these variables start to climb you may have a situation that will negatively impact availability. If queue depth grows more jobs may help at the risk of overloading the system's CPUs. The inverse of this is true also, if the system load grows too much lowering the number of jobs will lower the load at the possible growth of queues. System tuning is finding the balance by changing the number of jobs to keep the queues manageable while leaving system headroom.

Lessons of Sobig

During this implementation we did not manage to get AV scanners on the desktop. This became an issue for members of the team in the summer of 2003 with the release of the Sobig.f virus. Not only as a concern for the performance of the scanners on the iPlanet/SunONE messaging server but as part of the campus's general response to this major virus.

During the crisis we had to revisit scanning on the desktop as part of campus wide AV policy. In part because this virus had infection vectors other than email, our organization found a need to implement a campus wide desktop solution. While this change in itself was not significant, the impact was.

In a review of the events around the summer of 2003 Sobig virus outbreak, we examined our mail scanner logs. The result of our investigation was surprising. We found that we had cut the number of virus infected emails originating from on campus by over half. While we had expected a change in our stats we had not anticipated such a large drop. The lesson of multiple layers of security improving our security was driven home by this very quantitative result.

There are several areas still open for improvement. Two of the still pending projects are implementing a system to drop notification to the end user when we have very large virus load and a way to simply change the notification. The first of these two has already been partly done (in the wake of sobig.f). The major limitation is the lack of a documented way to drop a message from the mail server. A work around using the held function has been found, but would require a second utility to clean held messages. The current solution for this function is being handled by the spam scanning system we have in place. Spam rules on the scanners are set to mark high flow virus messages as spam. This allows people to selectively block high flow messages about virus infections as an opt-in function. This temporary function using the spam scanner may end up being our permanent one. The simple change of the notification message is not seen as a high priority, and may be shelved indefinitely.

Conclusion

The system developed through this process has given us a stable solution for virus scanning. Not only does it provide reliable and efficient scanning in all but the most severe message storms, but it also requires virtually no maintenance to keep it running. By doing all the research and testing we were able to add this security function in without changing the mail system's performance or administrative overhead.

This document has provided a guide to implementing an iPlanet/SunONE messaging server with AV that maintains availability. Providing an example methodology of adding security to an existing system with minimal impact on the function of that system.

Appendix

AV engine performance testing scripts:

In the same directory both of these files and a real infected file if possible or the eicar file.

spin.ksh

```
#!/bin/ksh
let test=1
# set the number of test cycles
let cycle=10
./vreset
while (( test <= cycles )); do
timex /usr/local/bin/uvscan -m /tmp --clean --unzip infected-file.exe
./vreset
let test="test + 1"
done
```

vreset.ksh

```
#!/bin/ksh
cp infected-file infected-file.exe
```

AV scripts and engine performance testing scripts/environment:

In the same directory both of these files and a real infected file if possible or the eicar file.

spin-script.ksh

```
#!/bin/ksh
let test=1
let cycles=11
export INPUT_FILE=/path-to-testing-dir/scantest/infected-file.exe
export OUTPUT_FILE=/path-to-testing-dir/scantest/infected-file.exe
export NAME=test
export INPUT_TYPE=Application
export MESSAGE_HEADERS=/ path-to-testing-dir/scantest/msg-headers
export INPUT_HEADERS=/ path-to-testing-dir/scantest/msg-headers
./vreset
while (( test <= cycles )); do
timex /path-to-testing-dir/scantest/multiscan.sh
./vreset1
let test="test + 1"
done
```

NAI DAT File Update Script

update_dat.ksh

```
#!/bin/ksh
# Based on the FreeBSD script and modified for solaris
#  by eric straavaldsen
#
# needs the Sun Packages for gnu-tar and ncftp
#   SFWgtar & SFWncftp
# setup for ----------- iPlanet AV Scanner by
#                       e9s 9/23/02
#    updated with more comments and a few fixes.
#                       e9s 9/30/02
#
UVSCANDIR="/usr/local/uvscan"
DAT_SITE="ftp://ftp.nai.com/pub/datfiles/english/"
DAT_FILES="clean.dat internet.dat names.dat scan.dat readme.txt"
TMPDIR="/tmp/datupdate"

AWK="/bin/awk"
CP="/bin/cp"
ECHO="/bin/echo"
GREP="/bin/grep"
GTAR="/opt/sfw/bin/gtar"
FTP="/opt/sfw/bin/ncftpget"
MKDIR="/bin/mkdir"
MV="/bin/mv"
RM="/bin/rm"
SED="/bin/sed"

progname="update_dat.sh"

while getopts vf: arg; do
        case $arg in
        v)      verbose=1 ;;
        f)      dat_tar="${OPTARG}" ;;
        esac
done

${MKDIR} ${TMPDIR}

if [ -n "$dat_tar" ]; then
        if ! (${GTAR} -x -C ${TMPDIR} -f $dat_tar readme.txt >/dev/null); then
                ${ECHO} "$progname: unable to extract readme.txt"
                ${RM} -rf ${TMPDIR}
                exit 1
        fi
else
        # Fetch the ReadMe file to read the latest version of the DAT files.
if ! (cd ${TMPDIR}; ${FTP} -V ${DAT_SITE}readme.txt >/dev/null); then
                ${ECHO} "$progname: unable to fetch ${DAT_SITE}readme.txt"
                ${RM} -rf ${TMPDIR}
                exit 1
        fi
fi

curver=`${AWK} '/DAT Version/ { print $4; exit }' ${TMPDIR}/readme.txt | ${SED}
                      -e 's/^.*\([0-9][0-9][0-9][0-9]\).*$/\1/'`
oldver=`${AWK} '/DAT Version/ { print $4; exit }' ${UVSCANDIR}/readme.txt |
                      ${SED} -e 's/^.*\([0-9][0-9][0-9][0-9]\).*$/\1/'`

if [ $curver -le $oldver ]; then
        if [ -z "$verbose" ]; then
                ${ECHO} "$progname: VirusScan DAT files are current ($oldver)"
        fi
else
        if [ -z "$dat_tar" ]; then
                dat_tar=${DAT_SITE}dat-$curver.tar
                if (cd ${TMPDIR}; ${FTP} -V $dat_tar >/dev/null); then
                        dat_tar=${TMPDIR}/dat-$curver.tar
                else
                        ${ECHO} "$progname: unable to fetch $dat_tar"
                        ${RM} -rf ${TMPDIR}
                        exit 1
                fi
        fi

        ${GTAR} -x -C ${TMPDIR} -f $dat_tar

        # Backup old dat-* tar files.
        if [ "`${ECHO} ${UVSCANDIR}/*.tar`" != "${UVSCANDIR}/*.tar" ]; then
                for file in ${UVSCANDIR}/*.tar; do
                        ${MV} -f $file $file.old
                done
        fi

        # Backup old DAT files.
        for file in ${DAT_FILES}; do
                file=${UVSCANDIR}/$file
                if [ -f $file.dat ]; then
                        ${MV} -f $file $file.bak
                fi
        done

        # Copy new DAT files into place.
        for file in ${DAT_FILES}; do
                ${CP} -f ${TMPDIR}/$file ${UVSCANDIR}/$file
```

```
     done
       ${CP} -f $dat_tar ${UVSCANDIR}
       ${RM} -f ${UVSCANDIR}/*.old
       ${ECHO} `date` Successfully updated VirusScan DAT files to $curver.
fi

${RM} -rf ${TMPDIR}
# test the scanner function
/usr/local/uvscan/uvscan /usr/local/uvscan/eicar.com 2>/dev/null 1>/dev/null
retcode=$?
if [ $retcode = 2 ]; then
 echo " Driver integrity check failed. Possible scanner problem"
fi
if [ $retcode = 6 ]; then
 echo " A general problem occurred. Possible scanner problem"
fi
if [ $retcode = 8 ]; then
 echo " Driver could not be found. Possible scanner problem"
fi
if [ $retcode = 15 ]; then
 echo "The scanner's self check failed; it may be infected or damaged"
fi
if [ $retcode = 13 ]; then
 echo "The scanner's test succeeded."
fi

exit 0
```

<u>imta.cnf and job_controller.cnf examples</u>

imta.cnf

```
!
! the 6 at the end will be adjusted for performance
! it maintains the number of threads for the conversionn pool
! also make sure that the job_controller.cnf is also updated
! to reflect the number of threads
! conversion
conversion subdirs 20 maxjobs 3 POOL CONVERSION_POOL threaddepth 2
conversion-daemon

!
! conversion_ext
conversion_ext subdirs 20 maxjobs 5 POOL CONV_EXT_POOL threaddepth 2
conversion_ext-daemon
```

job_controller.cnf

```
[POOL=CONVERSION_POOL]
job_limit=3
!
[POOL=CONV_EXT_POOL]
job_limit=5
!
```

[i] http://www.icsalabs.com/html/communities/antivirus/certifiedproducts.shtml (12/2003)
[ii] http://www.westcoast.com/checkmark/index.htm (12/2003)
[iii] http://www.milter.org/ (12/2003)
[iv] http://www.eicar.org/anti_virus_test_file.htm (12/2003) The eicar test file allowed us to do test of effectiveness of scanning with out having to use a real virus. Since it also comes as a zip and com file we could test our scanner's ability to check within zip files.
[v] http://www.mozilla.org/projects/mstone/ (12/2003) The current home of the mailstone stress and performance tool.
[vi] Orca is an open source system performance graphing tool, for details please see http://www.orcaware.com (12/2003)

[vii] An interesting function we found was that the time it took to do the first scan in a test was 10-20% more than subsequent scans. As long as we scanned more frequently than once a minuet we did not lose this gain. After looking into the issue it was driven by the memory caching of the system of I/O.
[viii] http://docs.sun.com/source/816-6092-10/scan_sh.tar.gz (12/2003) also see David Evans GSEC certification paper "MTA based Virus Scanning with Sun One Messaging and Sophos" http://www.giac.org/practical/David_Evans_GSEC.doc (12/2003) for a good step by step and tools for Sophos.
[ix] We chose to scan the message and clean if possible and if not possible to the delete it. We wanted to be as non-invasive as we could be.
[x] For watching system load on a Sun server I strongly suggest that you use the mpstat tool. This will most accurately show how much of your processor performance you are using. Tools like top and prstat show you the system load, and while is useful may not show you how much of your CPU time you are using. Beyound idle time you should watch to make sure your system time does not become more than 1 part in 3 of your user time. More than 1 part in 3 of system time is reason to look for resource contention.