

Models of Viral Propagation

Gabor Szappanos 2000-07-17

With the worst of the LoveLetter threat behind us, it's a good time to take a closer look at the incident, in order to better understand what happened and why. I am not going to calculate the gross damage caused by the worm. Rather, I will employ some simplified mathematical formulae and conclude what should be done, and by whom, in order to avoid similar attacks.

According to the data collected by the International Computer Security Association (ICSA), Loveletter has the dubious honor of being the fastest-spreading computer virus in history.

In the early era of computing, a successful virus like Form could become the world's most prevalent virus within 3 years of its first appearance. Five years later, in 1995, Concept took only four months to climb to number one. Last year, Melissa took a scant few days to reach the top of the virus list.

All this is nothing compared to Loveletter, which became the most widespread virus ever within four hours of its first encounter. Within that time, it had already infected over a million PCs.

I will discuss the incident from the point of view of an epidemiologist tracking the spread of a contagious disease in a population. I will use the simplest models, assuming homogeneous virus propagation, which means that each infected computer can infect any other computer in the world with equal chance.

This assumption did not hold true for traditional boot and file viruses, where propagation followed a more hierarchical graph model in which infection tends to propagate to computers in the neighborhood of the infection source. However, in LoveLetter's case, the infection spreads to addressees in an infected user's address book; as e-mail correspondence tends to be global and not restricted to local users, the propagation of an e-mail worm can be considered homogeneous.

The life cycle of the worm is divided into 2 stages. In the first stage it is still unknown to virus scanners. The second stage occurs after the virus signatures for the scanner have been updated.

In the first stage, the following equation describes the number of infected systems:

$$N(t) = e^{(g-a)\frac{t}{\tau}}$$

In our equation, t is the elapsed time, g is the generation rate (the average number of infected e-mail messages sent out in an infection cycle), a is the absorption rate (the number of worms that die out in an infection cycle and will not cause any further infections), and τ is the cycle time of an elementary virus spread. To put it simply: in each infection cycle, g copies of the virus are being sent out, a of which are not successful, and do not cause further infections on the target computers.

After the new virus's appearance, antivirus companies learn about it from user feedback and soon, t_0 time after then, they are ready with the virus signature update that detects and cleans it.

After the virus scanner updates are out, it is reasonable to assume that they are installed on p percent of the infected PCs (which of course means that the worm "dies" immediately on these PCs) leaving $1-p$ ratio remaining infected.

This is also a realistic assumption. If there is something system administrators learned from past experience, it that virus updates should be installed promptly in case of such an incident.

The propagation in the second stage starts with the $(1-p)$ fraction of the total infected PCs at the moment the virus signature update was released.

Another difference compared to the first stage is that the multiplication number is also decreased by the same factor as the update is installed on the possible target PCs.

The infected PC count in this stage is described as follows:

$$N(t) = (1-p) \cdot e^{(g-a)\frac{t_0}{\tau}} \cdot e^{(1-p)(g-a)\frac{t-t_0}{\tau}}$$

Now let us see what can be done to reduce the total number of infected systems, and who has to take these steps. The rest is simple mathematics.

How can we decrease the number of the virus population that follows an exponentially increasing law? By decreasing the exponent, if the virus multiplication factor ($g-a$) is higher than 1, the virus will spread at an exponentially increasing rate. If it is less than 1, it will die out. Assuming typical users with typical address book sizes, this multiplication rate could be well over 10.

Just a quick comparison: Nuclear chain reactions are driven by pretty much the same differential equations as virus propagation. 14 years ago in an incident in Chernobyl, the multiplication factor was slightly above 1.2. You may have heard about its outcome. Exactly the same is happening to all of us nowadays. In the last half-year, two such "nuclear" explosions have shaken the Internet: the Melissa and Loveletter mailbombs.

According to ICSA's survey, the damage caused by the first is estimated to be about \$100 million USD, while Loveletter's damage is above \$700 million USD. This cannot be a tolerated risk of the Internet and e-mail usage: similar threats must be eliminated in the future.

A trivial solution to decrease the virus population is to increase factor p - that is, to buy more virus scanners and install them on more PCs. It is true that this solution decreases the exponential, additionally ensuring that the population will die out faster in the second life stage of the worm. The only problem is that simply buying more scanners does not decrease the infection count in the first life stage of the virus. Moreover, other factors could reduce virus threats apart from forcing users to spend more money on virus scanners. However, this requires writing better virus protection tools.

The virus life cycle \square approximately equals the frequency with which an average user reads his/her mail.

Just for comparison: for an average boot virus the life cycle time was about the time needed to transfer the virus manually from one PC to another on floppy and activating it on the new PC. It could be estimated at roughly one week, as the user had to walk from one PC to another.

The cycle time is about one hour for an e-mail virus. No wonder Loveletter could spread like fire: it is pure mathematics. And there is little hope it can be reduced (unless users are ordered to check their mail only once a day). Indeed, trends are just the opposite: Internet users are getting to enjoy to online presence and are not going to give it up.

We should investigate the other parameters for better protection.

Introducing some blocks in the mass-mailing capabilities of Outlook can decrease the number of worm copies sent out. Microsoft quickly released a security patch to decrease this factor. They call it Object Model Guard as it attempts to block mass mailing.

In fact, all it does is just checking if the user address book is opened while Outlook is driven by an external program using Automation (just like Melissa and Loveletter did), and blocks these attempts.

We may argue about the use of this patch - several major companies expressed their disagreement because they use home-brew macros and internal distribution lists to mass mail most of their e-mail traffic overnight, but reducing g is clearly Microsoft's responsibility.

The number of absorbed worms is another story.

Factor a comes from several additive components. The worm is blocked if the mail arrives to a PC that is not running an e-mail client capable of activating it (a_1) or if there is virus protection installed that detects it (a_2). Clearly we cannot do much about a_1 as users cannot be forced to use simplified e-mail clients like PINE (which is a very tempting idea as far as security is concerned); considering the current trends, Outlook will remain the dominant e-mail client.

There could be a third hidden option to increase this factor: the users should accept the rule of not running any attachments before thorough checking.

Recent experiences proved that it is not going to happen. Virus experts expected that the public learned the lesson of the Melissa incident and people would not open e-mail attachments before thoroughly checking them. Nothing could be farther from the truth, as it seems the average user has forgotten everything he/she heard in the news in the past few months, and still happily launches any executable attachments.

Like it or not, the users will not practice safe e-mailing - unless they are forced by their employers. Major companies should accept strict internal rules regarding e-mail security and forbid the execution of e-mail attachments.

On the other hand, there is a great deal of work for us as far as a_2 is concerned.

The virus scanners should have been able to block the new virus. Let us be honest: Loveletter was not a revolutionary new virus; any decent heuristic engine should have caught it. It did not contain any new ideas; in particular, the mass-mailing procedure is practically identical to that of Melissa.

Ladies and Gentlemen, get to work and increase a_2 as much as possible. Decreasing the t_0 reaction time of the AV companies can decrease the total number of released viruses.

Most major antivirus companies feature automated analysis and signature extraction for incoming new virus samples. While this could reduce the reaction time to about one hour, there is not much hope to decrease it further.

In this matter the AV community did not fail during the Loveletter case: the signature updates were released as fast as humanly possible.

The formulas provided here are approximate, and some assumptions may not be 100% valid. However, this strictly mathematical approach clearly points out which factors may be improved in order to avoid Loveletter-like mass infection.

During the incident and in the following week the media and the AV companies pushed that the best solution to avoid similar attacks is to defend more PCs with virus scanners. Stock markets also reacted in the same manner, with increasing stock rates for the major AV companies. Everything seemed as if it was in perfect order and AV developers did a great job releasing virus signature updates a couple of hours after the first incident.

In reality, the Loveletter case proved that the current antivirus solutions are in a crisis, being incapable of avoiding similar incidents. Just a quick look at the ICSA shows that at the time the new virus signatures were out, the worm was already widespread all over the world. Current reactive antivirus techniques with 2-hour typical response time simply cannot stop similar attacks in the future.

The solution is to utilize more efficient generic virus detection methods in addition to their traditional signature scanners. There are at least two solutions to detect unknown viruses. Interestingly enough the major antivirus products use them both very efficiently for the detection of boot and program viruses (detecting over 90% of new variants) but rather inefficiently for macro and script viruses.

The first method is heuristic scanning, which investigates the instructions in the sample looking for instructions that are characteristic to viruses. If such instructions are found, the sample is suspected to contain a virus.

The second method is using a code emulator. This creates a virtual PC environment and examines the code of the sample file instruction-by-instruction, emulating the changes caused by the instructions in the emulated environment. If the environment is getting to look like it has been infected by a virus, the sample is suspected to contain a virus.

The major problem with generic methods is that they produce false alarms, suspecting infection in innocent programs. The public has to make a decision: what is worse, suffering occasional false alarms, or wasting millions of dollars on regular virus incidents?

I know my answer.

Relevant Links

[LoveLetter Variants](#)

by Sophos Anti-Virus

[Virus Glossary](#)

by McAfee

[ICSA.net](#)

[Cheetah Genetics](#)

by Robert Irion

[Link 3](#)

by Author 3

[Link 4](#)

by Author 4

[Link 4](#)

[Privacy Statement](#)

Copyright 2006, SecurityFocus