A randomized defence of virus throttling

Miranda Mowbray, Ganesh Ananthanarayanan, Anthony Joseph

HP Laboratories
HPL-2008-135

Keyword(s):
virus throttling, security

Abstract:

This paper gives a simple example of a defence against a worm attack which is a randomized combination of pure strategies, and which is superior to all of the pure strategies that it combines. Although it was developed to defend against an attack on virus throttling in a specific network device, both the attack and the defence are rather generic, and may be applicable to other areas.

External Posting Date: October 21, 2008 [Fulltext] Approved for External Publication
Internal Posting Date: October 21, 2008 [Fulltext]

Presented at ARCS (Workshop on Adaptive Resilient Computing Security), London. September 30, 2008

A randomized defence of virus throttling

Miranda Mowbray (HP Labs Bristol), Ganesh Ananthanarayanan and Anthony Joseph (UC Berkeley)

Introduction

In ARCS 2007, one of the authors of this paper described some useful properties of a 13th century randomized election protocol, and suggested that although randomized protocols are horrible to debug, they are nevertheless worth considering for use in network security. This is because the randomization gives more room to play with than deterministic protocols can have. As a result it enables some security properties which are infeasible for deterministic protocols.

Following up this idea, this paper gives a simple example of a defence against a worm attack which is a randomized combination of pure strategies, and which is superior to all of the pure strategies that it combines. Although it was developed to defend against an attack on virus throttling in a specific network device, both the attack and the defence are rather generic, and may be applicable to other areas.

Randomized strategies – an example

A standard result in Game Theory is that a randomized strategy may beat all pure (that is, deterministic) strategies. Consider the two-player game *scissors-paper-stone*. Strategies for this game are rules which tell players how to play in the current round, given what both players have done in all previous rounds. If your opponent knows your strategy, in most cases she can beat you – for instance, if she knows you're your strategy will lead you to play *scissors* in the current round given what has gone before, she will choose to play *stone* in the current round. However, if your strategy is the randomized one in you choose *scissors*, *paper* or *stone* with equal probability on every round, then you and your opponent are expected to draw.

One approach to the general observation in game theory that an opponent who knows your strategy may find it easy to beat you is to try to keep your strategy secret and hard to guess or deduce. An alternative approach is the one used in this paper, which is to use a randomized strategy with the property that even if your opponent knows your strategy and uses this information to determine her own best strategy, she is still not expected to win.

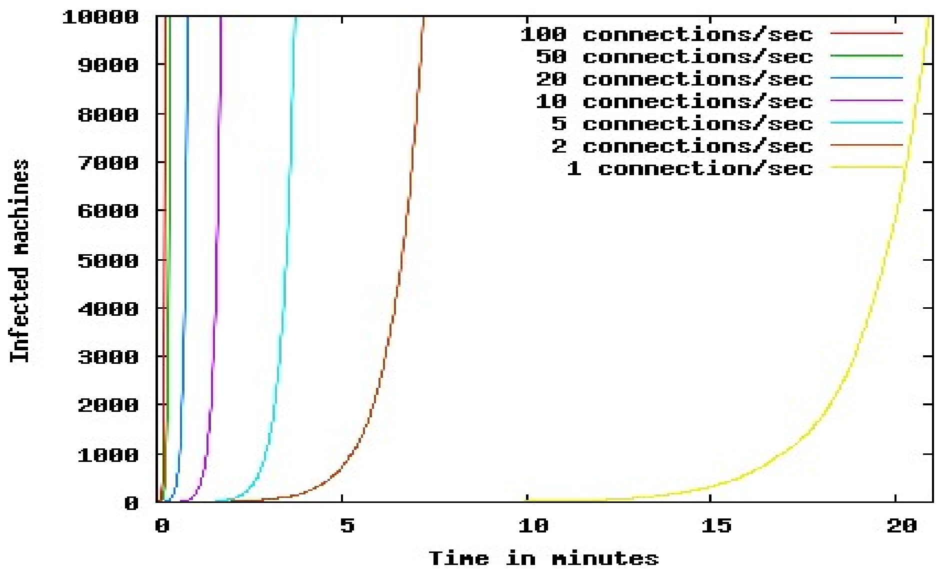
Virus Throttling

The idea of virus throttling, a network security technique invented by Williamson et al and presented in an early ARCS workshop, is as follows. Virulent worms propagate by connecting to a large number of machines in a short time. Therefore, to slow down the spread of such a worm, virus throttling prevents a machine from making connections to more than *T* different machines per second, where *T* is a fixed threshold. Requests for connections that are not allowed in that second are queued until the virus throttling mechanism will allow them to take place. Virus throttling has been shown to be effective in practice at slowing down the spread of very virulent worms.

Virus throttling is only a delaying tactic. It does not in itself prevent the worm from taking over the network; it just slows down the worm's spread, to give administrators the chance to apply patches or other mitigation methods. If a machine requests a very large number of connections to different machines in a set time interval (possibly a different time interval than the second interval for throttling), a router with virus throttling enabled may decide that this is an indication that the machine is infected, and block the machine from making further connections (and the network security system may also take steps to disinfect it). This blocking can prevent virulent worms from taking over the network.

Since some perfectly legitimate network applications have behaviour that results in connections to a large number of connections to different machines in a short time interval, it may be wise not to automatically block a machine with this behaviour, but rather to quarantine it for a short time to allow a more careful investigation of whether it is infected, and to block further communications from the machine only if the investigation reveals that indeed it is infected.

The problem with this defence is that a cunning worm designer can set the number of different machines connected to by an infected machine to be the highest number which does not cross the threshold for throttling or blocking. The Figure below shows how quickly a worm making 100, 50, 20, 10, 5, 2 or 1 connections per second infects all the machines in a 10,000-machine network, under the assumptions that this rate is less than the threshold for virus throttling, and that each connection by an infected machine has a 1% chance of creating a new infection. As you can see, an unblocked infection that connects to only 10 different IP addresses per second can take over the whole network in under 3 minutes. Typical thresholds for virus throttling are above this rate.



A game-theoretic model

Let C be the number of different machines in a large network that an infected machine requests to connect to during a time interval, T the blocking threshold, and n the number of infected machines at the beginning of the time interval. Assume that each unblocked connection made by an infected machine has probability p of creating a new infection, and that once a machine is infected it stays infected unless it is blocked. We will ignore effects of the size of the network; once the number of infected machines becomes close to the network size it will become more difficult to create new infections, but an infection which has reached enough machines to make this effect significant is almost certain to take over the network in any case.

We can now give a simple model of the situation as a game with two players, the attacker and the defender. The defender can choose the threshold value T, and the attacker can choose the value of C. We will allow randomized strategies by the attacker or defender: let the probability that the attacker chooses C be $a(C)$, and the probability that the defender chooses threshold T be $d(T)$. Let $n(t)$ be the expected number of infected machines at time t. Then

$$n(t) = n(0).[\sum_{C,T:C\leq T} a(C).d(T).(1+pC)]^t$$

It follows that if $\sum_{C,T:C\leq T} a(C).d(T).(1+pC) > 1$ then the expected value of $n(t)$ tends to infinity as t grows – so in this case the attacker will succeed in infecting the whole network – and if $\sum_{C,T:C\leq T} a(C).d(T).(1+pC) < 1$ then the expected value of $n(t)$ tends to zero as t grows – so in this case the attack is expected to fail.
If $\sum_{C,T:C\leq T} a(C).d(T).(1+pC) = 1$ then the expected value of $n(t)$ is $n(0)$ for all t, so once again the attack is expected to fail to take over the network.

First consider the point of view of the attacker. The attacker would like to maximize $n(t)$, which is equivalent to maximizing

$$\sum_{C,T:C\leq T} a(C).d(T).(1+pC)$$
$$= \sum_{C} a(C) [\sum_{T: C\leq T} d(T). (1+pC)]$$

This can be achieved by setting

$$a(C) = 1 \text{ for } C=C_{\max},\ a(C)=0 \text{ otherwise,}$$

where $C_{\max}$ is a value of C for which $\sum_{T: C\leq T} d(T). (1+pC)$ is maximal. Thus, the attacker has an optimum strategy in which an infected machine connects to the same number of machines in every time interval.

Now consider the point of view of the defender. For each $C\geq 0$, write $b(C)$ for the probability that the defender will block an infected machine making connections to C others in a unit time interval. We have

$$b(C) = 1 - \sum_{T:C\leq T} d(T) \text{ if } C>0, \text{ and } b(0)=0.$$

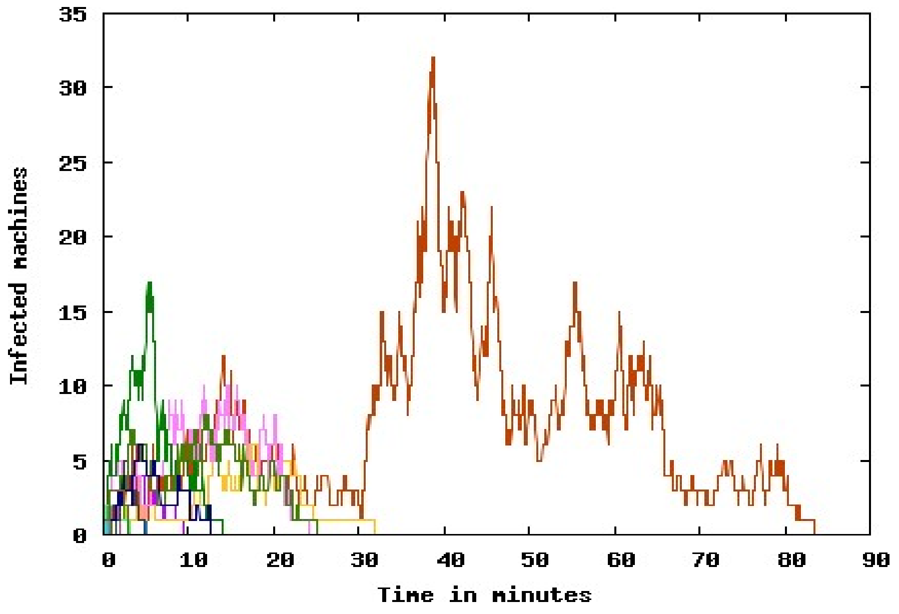
The defender would like to ensure that the attack is expected to fail, ie. that

$$\sum_{C,T:C\leq T} a(C).d(T).(1+pC) \leq 1, \text{ whatever values of } a(C) \text{ the attacker chooses.}$$

Moreover, for each C, subject to these inequalities, the defender would like to have the smallest possible values of $b(C)$, since the larger the value of $b(C)$, the larger the probability is of a non-infected machine being blocked if it attempts to make C connections in a time interval. Solving these conditions, the optimum strategy for the defender is a randomized defence in which

$$d(T) = p/[(1+pT-p).(1+pT)] \text{ for all } T>0,\ d(0)=0.$$

The figure below shows the result of 25 attacks with 1% infection rate, making 1 connection per second, under this defence. All die out within an hour and a half, and none infect more than 33 machines out of the 10,000 simultaneously.



Virulent attacks that make more connections per second also are expected to fail; although they typically infect larger numbers of machines initially than slower attacks, they have higher probability of being blocked and so typically die out more quickly.

Conclusion: when this defence is suitable

The drawback of this defence is the false positives that it will cause. It is important to minimize the disruption caused to legitimate users when their packets are blocked (by quarantining the packet and doing further investigation rather than immediately blocking the sending machine). Moreover, this defence strategy is not appropriate for dealing with infections with high infection rates, or systems in which it is common for uninfected nodes to make a large number of connections, as in these cases the false positive rate will be large.

Nevertheless, the defence is rather general. It is potentially applicable not only to virus throttling, but also to security-related actions triggered by traffic crossing a fixed threshold for, for example, the number of different ports scanned in a fixed time interval, the number of emails sent, or the percentage of failed connections. Other ARCS attenders may be working on systems susceptible to attacks for which a similar randomized defence may be useful. We hope that this paper will stimulate a discussion at the workshop about this.