Resilient Infrastructure for Network Security

Matthew M. Williamson
Information Infrastructure Laboratory
HP Laboratories Bristol
HPL-2002-273
October 11th , 2002*

E-mail: matthew_williamson@hp.com

Present day network security mechanisms are based on preventing attacks and responding to them as they occur. In the time before a response is implemented the attack is generally free to damage the system. Since responses are usually human driven, this time is long and the damage can be large. One way to minimise this damage is to create “resilient infrastructure”. This is infrastructure that automatically slows attacks so buying time for a human response. This paper argues the case for resilient infrastructure in network

* Internal Accession Date Only
Approved for External Publication

Resilient Infrastructure for Network Security

Matthew M. Williamson
HP Labs Bristol, Filton Road, Stoke Gifford, BS34 8QZ, UK
matthew_williamson@hp.com

Abstract

Present day network security mechanisms are based on preventing attacks and responding to them as they occur. In the time before a response is implemented the attack is generally free to damage the system. Since responses are usually human driven, this time is long and the damage can be large.

One way to minimise this damage is to create "resilient infrastructure". This is infrastructure that automatically slows attacks so buying time for a human response.

This paper argues the case for resilient infrastructure in network security.

1 Introduction

Present day network security mechanisms are based on preventing attacks and detecting and responding to them should they occur. However, while most attacks can be prevented (by patching machines and using perimeter defences) it is difficult to prevent all of them. As networks increase in scale and complexity, eliminating vulnerabilities becomes more and more difficult. Responses to attacks are generally human driven and are thus late and slow compared to the speed of attacks. With increasing network complexity the response will be even slower.

This means that using these two approaches together still leaves a vulnerability in the system: attacks that occur can run freely until a response can be mounted. Because the time to a response can be large, this freedom greatly increases the damage to the network, and thus the work required to cleanup.

This paper argues that this situation demands a different approach to security engineering. Rather than trying to design secure systems from scratch, the approach is to observe the existing system behaviour, and determine ways to improve that behaviour without changing the underlying system. This is akin to taming or domestication of a wild animal: not redesigning the animal, but manipulating it in different ways to make it more useful to the owner.

For the problem highlighted above, an approach to "tame" the system is to construct infrastructure that can automatically hamper and disrupt attacks as they occur. This will limit the damage caused, and buy time for a human response. Slowing an attack is simpler than stopping it, so it makes sense to use automatic computer responses to slow attacks, and human responses to stop them. Having resilience built into the infrastructure reduces the burden on prevention: if the damage from a potential attack is limited by the infrastructure, there is less risk associated with not preventing it. Having resilience will also help with system behaviour when under attack: the effect of the attack will be limited, allowing the system to continue to operate, albeit at reduced capacity.

A simple analogy for this approach would be that when a city is invaded, a simple but effective tactic by the retreating population is to remove the street signs. The population can then still use the infrastructure (for counter attacks etc.) but the invading army is confused and unable to use the same infrastructure as effectively. Removing the street signs does not stop the invasion, but hampers it while a response can be mounted.

This technology is most obviously applicable to large attacks on network resources, for example fast spreading viruses[1], denial of service attacks, zombies, misconfigurations etc., where the infrastructure could prevent the network becoming overloaded and reduce the impact of these attacks. Viruses that spread slowly and release a payload at some synchronised later time could also be countered: the slow spread gives much more time for a human response, and the fast attack could be mitigated by the infrastructure. Intrusions would be another attack where this approach could work. It is probably less appropriate for crafted cryptographic attacks, e.g. man-in-the-middle ssh attacks.

The rest of this paper describes in detail the limitations of present day prevention and detection-response mechanisms in a network security context. Resilient infrastructure is then defined and described. Two examples of different sorts of resilient infrastructure are then described, highlighting how this approach can both complement and take the pressure off existing security mechanisms. The final sections discuss some of the potential weaknesses of this approach and draw conclusions.

[1] While there are rigorous definitions of virus and worm (Cohen, 1994; Nachenberg, 1999), in this paper the two words are used interchangeably.

2 Prevention

All "prevention" approaches to network security are an attempt to reduce the number of vulnerabilities and the number of vulnerable machines. This can be achieved by scanning networks for vulnerable machines and patching them, by conducting penetration and patch cycles, or by simulating attacks from outside. An alternative way of preventing attacks is the use of defence in depth with firewalls, DMZs etc. to shelter machines from the ravages of the network outside.

Unfortunately, for prevention to be successful it needs to be complete or nearly complete. It is best for all the vulnerabilities to be removed, the firewall to only allow traffic that cannot attack the machines inside etc.. Getting complete or almost complete coverage is expensive and difficult. It is an 80/20 problem, with the cost of the last 20% being high.

There are a number of reasons for this, the first two being scale and complexity. Modern computer systems are big, and securing all those machines is difficult. If you make secure 90% of a network of 600,000 machines, there are still 60,000 vulnerable machines. A related issue is complexity. The computers on the network are no longer just servers and terminals, but servers, desktops, laptops, pdas, cell phones etc., all first class citizens and increasingly all with sufficient computing power and bandwidth to do damage. Just managing this diverse range of machines and users is a difficult task, while ensuring that known vulnerabilities are removed is close to impossible. The complexity also means that putting neat boxes (e.g. firewalls) around systems is more and more difficult. For example, many machines are now mobile, operating both outside and inside the corporate firewall. This means that protecting those machines is more difficult. Another example is in working practises. Peer-to-Peer systems such as Groove (Groove Networks, 2002) allow people to work together independent of organisation. It is hard to protect such systems with technologies such as firewalls.

A third issue is the demand of computer users for functionality or convenience that is often directly opposed to security. A good example would be the power of scripting languages in Microsoft Office applications. Those features allowed users to e.g. make their spreadsheets more dynamic but were also an opportunity for a variety of worms and viruses (Grimes, 2001). While security is important to most users of computer equipment, it is not generally their core business or reason for purchase. There are some customers e.g. military, for whom security is the most important criterion, but for the vast majority the functionality of the machine is more important. This means that there is an economic difficulty with highly secure systems, since they inevitably mean restrictions on functionality. If preventing vulnerabilities means losing functionality, then implementing such security will be an uphill battle.

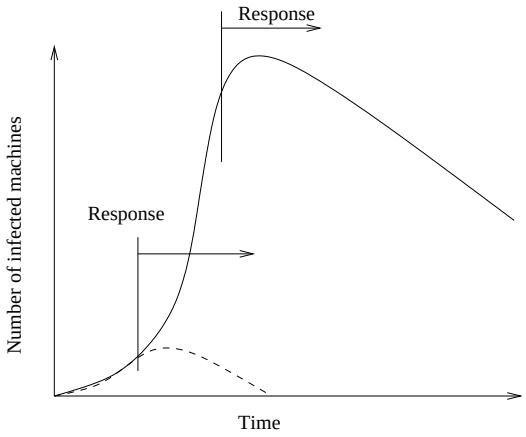

Figure 1: Number of infected machines against time for a virus attack. If the response is late (solid line) then the virus will spread much further before the response is implemented, meaning that the overall impact of the infection is large: many machines are infected and it takes a long time to clean up. If the response is quicker (dashed line) then the overall impact of the infection will be much less. Traditional responses to viruses require human involvement in the response; this makes them slow compared with the spreading speed of viruses.

A further issue is that security is a moving target. New vulnerabilities are announced regularly, requiring work to patch and upgrade systems. New applications are also continually developed requiring changes to firewall policies etc..

Finally, even if networks could be protected against all vulnerabilities, some attacks would persist simply because they exploit the correctly working system. For example, denial of service attacks that overload systems with many legitimate requests. It is hard to be resistant to this sort of attack without an enormous impact on functionality.

3 Detection and Response

As discussed above, it is impossible to eliminate all attacks, so other mechanisms are used catch the attacks that get through. These are usually grouped as detection and response.

While the speed of computers, networks, etc. has increased dramatically, the speed of responses has not changed significantly, making them relatively late and slow. This is a problem because it allows viruses to spread unchecked, denial of service attacks to proceed unthwarted and intruders to move around unchallenged for significant periods of time, and so do significant amounts of damage before a response is mounted.

The reason for the slow and late response is that humans are in the loop. An example would be the general

response to computer viruses. Figure 1 shows the typical progression of a virus infection for a fast spreading virus such as Code Red (CERT, 2001a). The response to a virus is generally to use a virus "signature". This is a fingerprint of the virus that can be generated once the virus has been noticed, isolated and analysed. These are fundamentally human driven processes which take time. Other response tactics are also slow, for example sending an email to warn about an email virus. During this delay the virus spreads effectively unhindered.

Another example would be Intrusion Detection Systems (IDS). Here a human has to sift through various alerts to determine if an intrusion is occurring and then respond. By the time the operator responds the damage could already have been done. The operator is more often concerned with assessing the damage and organising cleanup than with actual response.

It is not obvious how to speed up this response. Increasing the number of people working on the problem will not necessarily help and is expensive. As networks get larger and more complex it also becomes more difficult to mount a speedy response. It is also unclear how fast the response needs to be. Some theoretical viruses can spread through millions of systems in under a minute (Staniford et al., 2002). This is far too fast for a human response.

4 Resilient Infrastructure

The arguments in the previous sections can be represented by the diagram in Figure 2. The figure shows a representation of the space of vulnerabilities on the x-axis and the speed of response on the y-axis. Prevention techniques eliminate vulnerabilities are are represented by the area on the left. Attempts to increase prevention would move the border of that area to the right, but as discussed above it is difficult to move that line very far. Attacks that do occur (from vulnerabilities known but not removed, or from unknown vulnerabilities) are dealt with by responses. These are generally slow as represented on the diagram. It is hard to make responses faster.

The diagram thus makes the clear the vulnerability in the present approach: attacks that are not prevented can run riot until (much later) a response is mounted. The present solution to this problem is to balance the economic costs of closing the vulnerability with the risks of keeping it open. Unfortunately, increases in the scale or complexity of networks will only make this vulnerability larger, making prevention less effective and responses even slower.

The role of resilient infrastructure is to lessen the vulnerability, as illustrated in Figure 3. The aim is to mitigate attacks that have not been prevented until a human response can be mounted. The mitigation can be achieved by slowing down, hampering, confusing and disrupting malicious activity. If a firewall used to create a system that was "crunchy on the outside, chewy on the inside", then this infrastructure should make it "crunchy on the outside, treacle on the inside". The infrastructure should also collect data and alert the human, so that an effective human response can be made to finally deal with the problem.

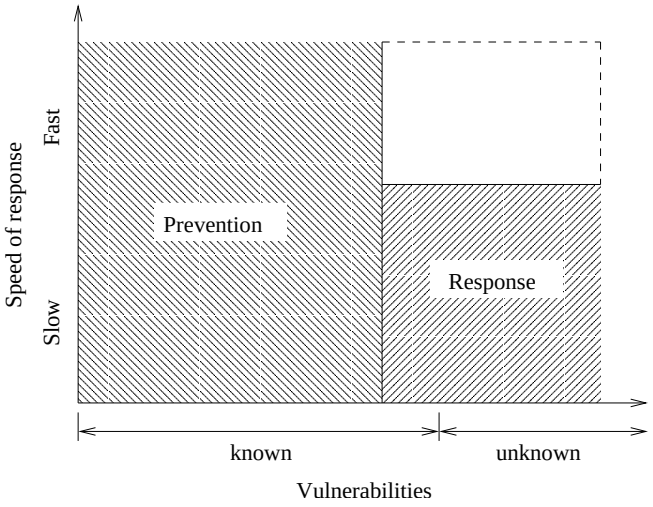

Figure 2: Representation of the space of vulnerabilities. The x-axis represents the space of vulnerabilities, and the y-axis the speed of response. Prevention techniques cover some proportion of known vulnerabilities (but not all of them, and cannot prevent attacks that are not known) and present day responses cover attacks that do occur but only slowly. The area that is left uncovered is in the top right hand corner, corresponding to a vulnerability to fast attacks. At present the economic costs of making that area as small as possible are balanced with the risks associated with leaving it uncovered. As networks get bigger and more complex the size of the vulnerable area will increase as it gets harder (and more expensive) to prevent or respond quickly to attacks.

As shown in Figure 3, resilient infrastructure complements prevention and detection-response. Since there is some technology addressing the vulnerability, there is less pressure on making the area as small as possible using the other approaches. If your infrastructure is resilient to certain attacks, that means less effort is required to prevent them. The human response can also be more relaxed. Both of these effects should reduce the cost of securing networks (balanced of course by any extra costs of resilience!).

There are a number of reasons for suggesting that attacks are hampered and not stopped. Firstly, it is hard to accurately detect attacks (as years of experience and research in Intrusion Detection Systems has shown (Newman et al., 2002; Axelsson, 1999)). If actions are takes to stop attacks on every alert, the rate of errors (false positives) will result in some actions being taken

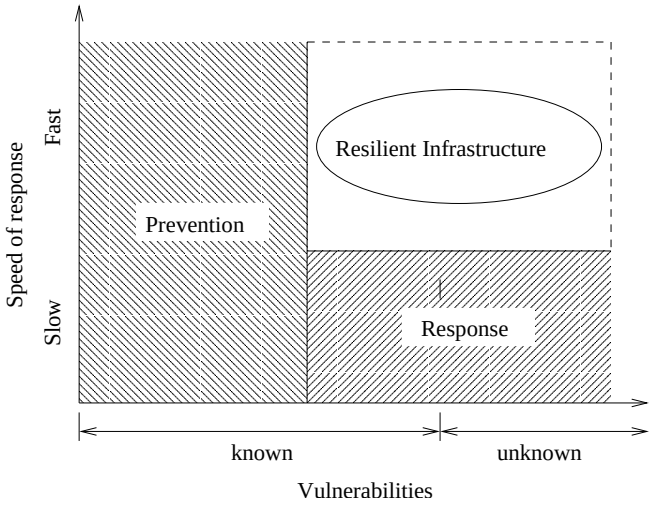

Figure 3: Diagram showing how resilient infrastructure can cover vulnerabilities in using only prevention and detection-response approaches. Having resilient infrastructure in place will not only reduce overall vulnerabilities, but also will take the pressure of the other techniques: fewer attacks need to be prevented, and the human response can be more relaxed. This is indicated by the smaller areas in the diagram.

by mistake with large reductions in performance. If however the response is benign, hampering or slowing the attack, then false alarms will cause much smaller drops in performance.

A second aspect is that it is often difficult to determine how to stop an attack, and stopping it is only one of the options available to the operator. The operator can use their common sense to determine the correct course of actions. Writing a computer program to perform this task would be very difficult, since it essentially requires artificial intelligence.

Resilient infrastructure thus allows a neat division of labour. Humans who are good at common sense and risk analysis but poor at responding quickly can be helped by an infrastructure whose role is to respond quickly but does not assume responsibility for decisions requiring common sense.

The following sections of this paper describe two examples of resilient infrastructure, showing how in each case the infrastructure addresses a need, and complements and takes the pressure off existing approaches.

5 Example I. Virus Throttling

As mentioned above, viruses spread quickly and without hinderance before a virus signature is developed. Virus throttling (Williamson, 2002) is a technique that slows the spread of viruses before the signature is available. It does this by placing a restriction on the network behaviour of a machine so that viral-like traffic is heavily delayed and slowed, but normal traffic is largely unaffected.

The technique is based on the observation that a machine that is infected by a virus will contact (and attempt to infect) as many machines as possible, as fast as possible. This is in contrast to an uninfected machine that makes outgoing connections at a lower rate, and those connections are locally correlated—the machine is more likely to re-connect to a recently contacted machine than a different one. This observation is born out for a wide variety of computer types (server, desktop machine etc.) and for different protocols (see Williamson (2002); Heberlein et al. (1990); Hofmeyr (1999)).

Given this observation, a mechanism that will allow normal traffic but prevent viral traffic is to implement a limit on the rate of connections allowed to "new" machines. "New" can be simply determined as being "in the last n connections made", where tests show that n is generally a small number (5–10). Such a simple detection mechanism is bound to make mistakes, so rather than drop connections that are to new machines, the idea is to delay them. The delay is organised so that the quicker and more uncorrelated the traffic, the more it is delayed. Occasional detection errors result in small delays, but virus-like behaviour is heavily delayed. Overall, the rate of connections to new machines is always less than a certain allowed rate, ensuring that viruses can only spread slowly.

Initial tests (Williamson, 2002) have shown that for a variety of different protocols the allowable rate can be as low as 1 connection per second. This appears to have negligible effect on normal behaviour (e.g. around 1 in 100 normal requests delayed by 1 second, 1 in 500 by 2 seconds), and is not particularly noticeable when implemented. Since the propagation of the virus is limited to a fixed rate, the effect on the virus propagation depends on its unrestricted rate. For viruses such as Code Red and Nimda (CERT, 2001a,b) which made 200–500 connections per second, the virus should be slowed by factors of 200–500. The throttle also makes it easy to detect viral-like activity (by monitoring the number of delayed connections), so it is possible to collect data on the virus and alert a human, so aiding the speedy generation of a signature.

In its present incarnation, the throttle acts more to protect the network of machines than to protect the individual. This is because it does not stop the virus destroying the individual machine but does restrict its propagation. While this is unfortunate for the machine, it does limit the overall damage to the network.

The throttle, if implemented widely, would create an infrastructure that is resistant to viruses. Normal (correlated) traffic will pass relatively unhindered, but viral traffic will be delayed. Having this resilience built into the infrastructure will take the pressure off the need completely remove vulnerabilities, and in fact also aid

the slower human response.

Other research efforts that are related to this include the idea of behaviour blocking (Messmer, 2002), and some work on preventing machines participating in network attacks by Brushi and Rosti (2000).

6 Example II. Responsive IDS

Intrusion detection systems are much studied and there is considerable controversy over their use and value (Newman et al., 2002). They fall into two types, those that use signature libraries, and those that attempt to learn the network behaviour and detect anomalies. The signature based systems suffer from the same problems as anti-virus signatures in that intruders can intrude freely until the signature is found and distributed. The anomaly detection systems attempt to solve a difficult problem, characterising the "normal" behaviour of a system and detecting differences from that normal profile. The problem is that computer systems are complex and change naturally over time: new software is installed and used, new users have slightly different work practises, old users leave etc.. Thus the profile needs to change over time too. This means that it is inevitable that the system will make mistakes, either missing real intrusions, or classifying a normal activity as an intrusion. This latter error is known as a false positive, and results in a waste of the operators time. Increasing the sensitivity of the system to real intrusions usually increases this false alarm rate too.

The irony is that even if an intrusion is correctly detected, by the time the operator does something about it, the damage has been done e.g. the web site is defaced, credit numbers stolen etc.. The job of the operator is more often damage assessment and cleanup than response. In addition the responses often cause problems in the future. For example one response is to drop traffic from a particular IP address after an attack has been detected from that address. This is fine, except that IP addresses are often dynamically assigned, so that traffic from that same address 1 hour later could be legitimate, but the system would deny it.

What is needed is a system to automatically and quickly react to intrusions as they occur. One such system was proposed by Somayaji and Forrest (2000).

Somayaji's system (implemented as part of the linux kernel) monitored the system calls made by running processes, looking for anomalous sequences, which could correspond to an attack. If such a sequence was detected, the execution of that program was slowed by inserting delays between the system calls proportional to the number of anomalies recently detected. This (like the virus throttling) has the effect that a program that is running normally is not delayed, false positive errors result in a slight loss in performance, and programs on which intrusions are attempted are heavily delayed. In addition, since his system detects and responds to anomalies, it could also respond sensibly to misconfiguration and other common network problems (Somayaji, 2002).

Somayaji presented a large variety of results on the performance of his system (see Somayaji (2002)). His system was effective against a variety of different exploits and misconfigurations, giving delays that ranged from tens of seconds to days. He also developed a user interface that would allow a human to interact with the system to check when processes were delayed. Some exploits were difficult to detect and gave rather short or no delays. As far as normal interaction was concerned, the behaviour of the system was relatively unobtrusive, but varied with programs (e.g. the behaviour of emacs was much more difficult to learn than sendmail).

This sort of system is another example of resilient infrastructure. It hampers programs that are behaving abnormally, so limiting the damage caused by attackers and indeed other network problems such as misconfiguration. It also provides a mechanism for alerting the user when under attack.

As with the other two examples, improving automatic responses complements and reduces the strains on other security mechanisms. Having a system that automatically fights intruders means that vulnerabilities can be left open with less risk. It also means that there is less work for the operator: false positives result in minor performance hits that are probably not noticeable, as opposed to being alerts that the operator must process. However, the operator still has overall control over the definitive response.

7 Drawbacks

Having presented the case for resilient infrastructure, this section considers some of the criticisms of the approach.

Firstly, how effective can we expect resilient infrastructure to be? Since it aims to slow and not stop attacks will that be futile as attacks get faster? If the aim is to disrupt malicious traffic but not normal, will the effect of the response have to be so small so as not to disturb normal operation that the effect on malicious traffic will be virtually nothing? For example does a hacker give up after 5 secs, 50 secs, 500 secs?

The true answer to these questions is unknown. There has been so little work in this area that it is too soon to tell. However, that work is generally positive. The virus throttling looks to be effective against fast spreading worms such as Code Red and Nimda (CERT, 2001a,b). However some theoretical viruses such as the Flash worm (Staniford et al., 2002) would probably be too fast for it. The Flash worm has a large list of addresses of machines to attack and quickly propagates to the machines on the list. It is estimated that the worm could take less than thirty seconds to infect 10m machines. If that worm was

throttled, then the same infection would take about 100 seconds (Williamson, 2002). This is slower, but is still too fast for a human response.

The work on intrusion response (Somayaji and Forrest, 2000) is also a promising data point, where some intrusions could be delayed for hours. There were however some cases where intrusions were not delayed much, or not delayed at all. These occurred when the exploit looked very similar to normal activity, since the amount of delay was proportional to the number of anomalies in a short space of time.

To summarise, some of the early work in this area is promising, with hampering attacks having a large and positive effect of virus transmission and intrusion response. However, the early work has also shown that in some cases the direct approach is less successful and a more sophisticated approach is required. For example Somayaji's system only monitored system calls. If his system had correlated system call and network activity it is possible that detection (and therefore response) could be more effective.

A related question is how to test whether these ideas are effective or not. There are two parts to this: do the techniques work well individually, and do they work well as part of the bigger system (prevention, detection and resilience).

Testing the individual pieces requires two processes. The techniques have to hamper real attacks, and so should be tested on them. The techniques also have to not disrupt normal activity, which is best verified using user trials.

Determining how resilience fits into the broader security context is more difficult. One approach would be to pilot systems and evaluate them. Another is to build models of the security process (for example epidemiological models of virus spread and cleanup (Leveille and Williamson, 2002)) and use them to gain insights into the impact of new processes.

A second criticism that is levelled at any automatic response system is that it opens a denial of service vulnerability. If an application slows down when it behaves abnormally, all an attacker needs to do is to craft an input that will make the application slow down.

This is a fair criticism that should be carefully addressed in the design of resilient infrastructure. The problem might not be as bad as suggested, as it may be difficult to know *a priori* what interactions may cause an application to behave abnormally. In addition, because systems change over time, the input that caused an application to be abnormal at one instant might not work at another. In any case this criticism needs to be taken in context. Adding resilient infrastucture may introduce extra vulnerabilities, but if it mitigates many other problems then it may be worthwhile.

A final comment is that our computer systems are not really engineered at present for this type of approach. A resilient system is somewhat like a control system, it needs sensors to detect what is going on, and actuators to allow it to alter the state of the system. The sensing ought to be rich enough, and the control action ideally needs to have enough granularity so that the controller can have a smooth response.

Unfortunately our computer systems do not presently have many sensors and have even fewer actuators! Part of the challenge for this work is to determine what sensors are necessary to instrument our systems, as well as inventing various actuation mechanisms: aspects of computer and network architectures that can be modified to create benign responses.

8 Conclusion

Present day security practises for defences of networks can be divided into two main areas. Prevention to eliminate as many vulnerabilities as possible, and detection-response to recover from attacks that do occur. Unfortunately, for prevention to be effective it needs to be nearly complete which is expensive and practically unfeasible. Responses are generally human driven and are slow, which is a problem because attacks are fast, and if not reacted to quickly result in the damage being much greater than necessary. With the ever increasing scale and complexity of networks, these techniques are straining at the seams.

This paper argues that there is a need for systems that can automatically react to attack and problems to complement these other two approaches. By building quick response mechanisms that hamper and disrupt attacks as they occur, the damage caused will be less. More time will be available for a human response to finally deal with the problem and there will be less need to remove all the vulnerabilities as the infrastructure will be resistant to attacks.

The paper has illustrated this with two examples from anti-virus and intrusion detection, both of which have demonstrated the value of a quick response that delays malicious activity. Using virus throttling will create a network over which viruses cannot propagate quickly. While the individual machine might not be protected, the overall damage to the network from an attack will be much less. Using the responsive IDS system will allow individual machines to protect themselves from attack. Both systems defer final action on the problem to a human.

Possible drawbacks to the idea have been discussed showing that initial research results are promising and highlight that more work is required. Part of the problem is determining how to sense system behaviour and also how to respond to that behaviour.

If this approach is successful, then it should grow with two goals. Firstly to further reduce the load on hu-

man operators, and secondly to encompass more network problems. Our systems will be easier to manage if they are naturally resilient to faults and misconfigurations as well as attacks.

Acknowledgements

This paper was aided by discussions with Jonathan Griffin, Dirk Kuhlmann and Graeme Proudler.

References

Axelsson, S. (1999). The base-rate fallacy and its implications for the difficulty of intrusion detection. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, Singapore.

Brushi, D. and Rosti, E. (2000). Disarming offense to facilitate defense. In *Proceedings of the New Security Paradigms Workshop*, Cork, Ireland.

CERT (2001a). CERT Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL. Available at `http://www.cert.org/advisories/CA-2001-19.html`.

CERT (2001b). CERT Advisory CA-2001-26 Nimda Worm. Available at `http://www.cert.org/advisories/CA-2001-26.html`.

Cohen, F. (1994). *A Short Course on Computer Viruses*. John Wiley & Sons, New York.

Grimes, R. A. (2001). *Malicious Mobile Code: Virus Protection for Windows*. O'Reilly & Associates, Inc.

Groove Networks (2002). `http://www.groove.net`.

Heberlein, L., Dias, G., Levitt, K., Mukherjee, B., Wood, J., and Wolber, D. (1990). A network security monitor. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 296–304. IEEE Press. `http://seclab.cs.ucdavis.edu/papers/pdfs/th-gd-90.pdf`.

Hofmeyr, S. A. (1999). *A Immunological Model of Distributed Detection and its Application to Computer Security*. PhD thesis, Department of Computer Science, University of New Mexico.

Leveille, J. and Williamson, M. M. (2002). An epidemiological model of virus spreading and cleanup. Technical Report In Preparation, Hewlett-Packard Labs.

Messmer, E. (2002). Behavior blocking repels new viruses. Network World Fusion News. Available from `http://www.nwfusion.com/news/2002/0128antivirus.html`.

Nachenberg, C. (1999). Computer parasitology. In *Proceedings of the Virus Bulletin International Conference*, pages 1–26.

Newman, D., Snyder, J., and Thayer, R. (2002). Crying wolf: False alarms hide attacks. *Network World Fusion*. Available from `http://www.nwfusion.com/techinsider/2002/0624security1.html`.

Somayaji, A. and Forrest, S. (2000). Automated response using system-call delays. In *Proceedings of the 9th USENIX Security Symposium*, pages 185–197, Denver, CO.

Somayaji, A. B. (2002). *Operating System Stability and Security through Process Homeostasis*. PhD thesis, University of New Mexico, Alberquerque, New Mexico. Available from `http://www.cs.unm.edu/~soma/pH/`.

Staniford, S., Paxson, V., and Weaver, N. (2002). How to 0wn the internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium (Security '02)*. Available at `http://www.icir.org/vern/papers/cdc-usenix-sec02/`.

Williamson, M. M. (2002). Throttling viruses: Restricting propagation to defeat malicous mobile code. In *Proceedings of ACSAC Security Conference*, Las Vegas, Nevada. Available from `http://www.hpl.hp.com/techreports/2002/HPL-2002-172.html`.