Leveraging Good Intentions to Reduce Unwanted Network Traffic

Marianne Shaw
marianne.shaw@gmail.com

Abstract

We present a solution to reduce unwanted network traffic by enabling either side of a conversation to summarily terminate the conversation without the other endpoint's cooperation. Our work is motivated by the observation that many compromised endhosts on the network are well-intentioned but easily compromised; these machines are often compromised and their resources used to attack others. We argue that the good intentions of these endhosts can be leveraged to construct a control plane that ensures that, even when compromised, these well-intentioned machines only generate well-behaved traffic. This independently enforced control plane prevents an endhost from blatantly disregarding requests to cease traffic generation. The solution's viability rests upon its unobtrusive deployment. No extra mechanism is needed within the network as all enforcement is performed at the endhosts. Hosts are not restricted in their behavior except by the behavior demanded by their peers.

1 Introduction

Commodity PCs and broadband have enabled huge numbers of users to connect to the Internet. Once connected to the Internet, user-administered machines are bombarded with attacks aimed at gaining control of their physical resources. Compromised machines are used to propagate worms and viruses, participate in DDoS attacks, provide services such as spam relays or IRC servers, and be part of organized botnets. Users do not want their machines to be compromised and used for malicious purposes, but they do not have the knowledge or skill to prevent it. This work asks, can we leverage the users' non-malicious intentions to prevent their machines from being used to generate unwanted traffic?

This work is driven by several key observations. First, we accept that a machine will be compromised and attempts will be made to use it to generate malicious network traffic. Attackers use compromised machines to amplify their ability to inflict damage; we can inhibit their potential impact by reducing the benefits of incorporating these machines. Next, we believe that many user-administrators do not want their machines to be used to inflict damage, and they would be willing to thwart such activity if they could. There is benefit to preventing the injection of malicious traffic into the Internet, rather than trying to deal with it once it is inside the network. We can leverage users' good intentions to co-locate an enforcement mechanism with a host and use it to prevent the injection of certain traffic into the network. Finally, defining and identifying unwanted behavior is difficult and often subjective; two hosts may not classify the same traffic in the same way. Rather than propose a universal definition of good or bad traffic, we seek to provide a way for traffic recipients to request the temporary cessation of traffic that they themselves deem undesirable.

Our solution puts control of the network packet exchange between two hosts in the hands of both of the endpoints in such a way that each endpoint has complete control over it. We do this by constructing an independent control plane that is co-located with each well-meaning host. When one endpoint requests that the other temporarily stops sending traffic to it, the control plane prevents that second host from disregarding the termination request; all outgoing packets are immediately dropped at the generating host and never enter the Internet. This conversation "ripcord" is necessary because a compromised host may have an altered networking stack or operating system [11] which ignores all (if they exist) standard termination requests. As long as both hosts believe that the conversation is well-behaved, there is minimal impact on the conversation.

2 Approach

Computers connected to the Internet continue to be attacked and compromised. User-administered machines are especially vulnerable to compromise as they often run unpatched computer programs that allow attackers to capitalize upon well-documented software flaws. Symantec [23] found that several popular unpatched desktop operating systems were compromised within 1.5 hours of being connected to the Internet. Once compromised, the physical resources of a machine may be used to further propagate an attack or may be incorporated into a network of attack machines.

We seek to prevent well-intentioned machines from being compromised and used to amplify an attacker's ability to inflict damage on the Internet. To do this, we make it possible for a host to tell a machine from which it is currently receiving packets to temporarily stop sending packets, and to reasonably expect that the request will be honored. Thus, if a compromised host is sending network attack traffic, a receiving host can stop the incoming packets to protect itself. While this work focuses on enabling endhosts to request the cessation of an attack, this mechanism could be used by overloaded hosts to temporarily delay incoming traffic.

There are a few key requirements for achieving our goals. 1) Upon receiving unwanted network traffic, a host must be able to identify the source of the traffic to which it can send a termination request. Any host that voluntarily adopts our mechanism must therefore be prevented from spoofing its packets' source address. 2) We only honor requests to temporarily terminate an existing packet stream. We do not allow hosts to proactively blacklist traffic that they have not yet received; nor do we honor termination requests for packet streams that have been inactive for a long duration. 3) Only a recipient of unwanted network traffic can request that a packet stream be terminated; malicious hosts should not be able to use this mechanism to force a well-intentioned host into silence. 4) Our enforcement mechanism must be voluntarily adopted by endhosts. We cannot rely upon the introduction of new mechanism within the network itself. We cannot impose undue restrictions upon the network services used by the host simply to accommodate our mechanism. 5) Upon receiving a termination request, we must be able to terminate a packet stream without the receiving machine's cooperation. Once a well-intentioned machine has been compromised, rootkits [9] can be used to gain superuser privileges on a machine; the machine's networking stack and OS can then be modified, replaced, or subverted [11]. Typically, a machine's OS and networking stack enforce well-established "good behavior;" once these modules are compromised, a machine is able to blatantly disregard all standard networking conventions.

We leverage the good intentions of non-malicious users to co-locate an enforcement mechanism with each host. The mechanism itself must be independent, not network-addressable, and able to interpose on all traffic in to and out of the host. Provided it can meet all of these specifications, the mechanism itself can be implemented in either hardware, software, or a combination of both.

Although each mechanism only enforces our requirements for an individual host, in aggregate these mechanisms create an independent control plane. This plane ensures that all traffic that it allows to enter the Internet can be summarily terminated at a recipient's request. Recipients of unwanted network traffic now have a course of action by which they can protect themselves without requiring the cooperation of their ISP.

3 Design

Our control-plane enforcement mechanisms must ensure that potential victims can accurately identify their attackers from the offending packet stream, determine the validity of requests to temporarily stop an existing packet stream, and enforce valid network traffic termination requests without endhost cooperation. A combination of control plane signalling and our enforcement mechanism make this possible.

3.1 Control plane signalling

To leverage users' good intentions, our approach must be able to provide significant benefits without introducing a system administration burden on the users. Once a user allows us to interpose on all network traffic in to and out of their machine, all necessary information should be gleaned from the traffic that we observe. From the stream of packets, we must be able to identify the co-located host's unique identifier, the start and end of a conversation between two hosts, and a termination request.

Unique Identifier Ideally, the Internet would provide each host with a unique non-forgeable identifier that could be used to provide accountability for actions taken by a networked host. In practice, hosts have the ability to transmit arbitrary network packets; they can assume another host's identity by transmitting packets with spoofed source addresses. Additionally, rather than having a single assigned static IP address, many hosts dynamically acquire their IP address for a short period of time.

Accountability is a necessary element of our solution. A recipient of unwanted traffic must be able to identify and contact the host that sent the traffic; at the same time, that host should not be penalized for spoofed packets sent by other machines. We must ensure that well-intentioned hosts cannot send packets with spoofed source addresses, but we must also monitor the packets that our host did send to deny accountability for others' actions.

A host's unique identifier is therefore the source IP address that it used to send a stream of packets during a specific time period. This approach, while not perfect, is sufficient for our purposes because we only honor valid termination requests for active traffic streams.

Our enforcement mechanism can determine a host's unique identifier from the consistent source address of the packets that it sends. We can leverage events that signify an expected change in IP address to recognize valid IP address changes. For example, hosts that dynamically acquire IP addresses tend to exhibit lulls in their network activity before they acquire a new IP address. Alternatively, if the control mechanism is directly connected to the host's network card, it can detect when a card is reset by the link going down. These events occur over a period of seconds. In contrast, many network attacks rapidly send packets with quickly changing spoofed source addresses; our mechanism should characterize these as spoofed packets and drop them.

We can prevent our host from being penalized for spoofed packets sent by other machines by tracking the packets that were actually sent by the host. Rather than log each individual packet transmitted, we can track the fact that we sent packets associated with a particular network conversation (defined below) during a certain time frame. When presented with a termination request for a packet that the host did not send, the enforcement mechanism simply discards the request. Because termination requests are only honored for active packet streams, the amount of state required is bounded by the number of currently active streams.

Defining a network conversation A network conversation defines both the criteria and the granularity that we use to track sequences of network packets; it dictates which packets will be dropped when a termination request is received. Hosts receiving unwanted traffic must weigh the cost of receiving those incoming packets against the cost of terminating the network conversation.

We can provide the ability for a host to summarily terminate a conversation for many different definitions of a network conversation provided we are able to uniquely identify the principals of a conversation from each packet sent by the host, identify the start and stop of a conversation by observing the packet stream, and identify the termination request associated with a conversation.

Conversation principals Traditionally, IP source and destination addresses have been the basis for identifying network conversations. Additional properties such as IP protocol and source and destination ports have been used to refine these principals. We can extend this set to include more coarse grained principals. IP prefixes could be used instead of IP addresses; thus, our enforcement mechanism can honor requests to drop all UDP port 666 traffic destined for 10.10.10.*. Alternatively, a host can request that it no longer be sent any TCP traffic.

Conversation start/stop The enforcement mechanism must know exactly what indicates that a conversation is active and inactive. Ideally, we can identify the start and stop of a conversation simply by observing the contents of network packets and maintaining internal state. For example, TCP uses explicit start, stop, and termination sequences for maintaining connections. We can use this protocol signalling to restrict and terminate wayward conversations; a prototype for TCP is outlined in Section 4.

However, for many conversations there is no explicit signalling indicating the conversation delimiters, and we must infer the start and stop of the conversation by observing patterns of network activity. Correctly inferring these endpoints can be difficult; although we can use the existence of network traffic between two principals to recognize that a conversation is active, for many long-lived conversations we cannot use the absence of network traffic to determine that a conversation has been stopped.

Termination requests Hosts require an explicit signalling mechanism for terminating a network conversation. In addition to indicating which network conversation is being terminated, these requests must either indicate or imply the amount of time during which packets must not be sent; we do not allow network conversations to be terminated indefinitely.

Certain protocols may have existing support for terminating a conversation; for example, TCP uses RST packets to reset a conversation. However, if we must infer active conversations based upon the existence of network traffic between two principals, it is unlikely that there will already be an existing explicit signal for terminating the conversation. To accommodate these ad-hoc definitions of conversations and enable hosts to reliably terminate them, it may be necessary to provide a new signalling mechanism.

Termination requests must demonstrate that the request is being sent by the recipient of the unwanted network traffic; spoofed termination requests should be discarded. The requesting host can be authenticated through the exchange of a large random nonce with the enforcement mechanism. If the nonce cannot be overlaid on top of a network conversation's existing protocol, then an explicit authenticating nonce exchange may be required. Once successfully exchanged, the nonce can be injected into a termination request to establish its authenticity.

3.2 Enforcement mechanism

To convince well-intentioned users to allow our enforcement mechanisms to interpose on their machine's network traffic, we must be unobtrusive yet effective at preventing their machines from attacking other hosts.

The enforcement mechanism cannot be bypassed or subverted by attackers The enforcement mechanism must interpose on all traffic in to and out of a machine, and it must remain completely isolated and independent from that machine. If the enforcement mechanism is not independent, when the host machine is compromised the attacker can simply "turn off" all packet-restricting components. Incapacitating the enforcement mechanism should require physical access to the host machine to prevent it being silently disabled by anonymous attackers.

The enforcement mechanism must actively participate in each conversation that it may need to forcibly terminate. This work aims to reduce attack traffic that is generated by a compromised host; in this scenario, *both* sides of the enforcement mechanism are controlled by the attacker. If the enforcement mechanism does not inject itself into a packet stream, the compromised machine can collude with an external attacker to prevent a conversation's termination.

Actively injecting a nonce into a packet stream enables the enforcement mechanism to independently authenticate an endpoint. Only hosts directly on the path taken by outgoing network packets will be able to reliably establish, maintain, or terminate a conversation.

The enforcement mechanism cannot be undermined by replaying a previous conversation through the mechanism This is especially important as many hosts acquire their IP addresses dynamically; an attacker could try to replay a previous conversation to inflict damage on the host now allocated a specific IP address. Therefore, the enforcement mechanism must require proof of "liveness" for all conversations flowing through it. The nonces used to authenticate endpoints should be randomly generated at the time they are needed.

The enforcement mechanism can be deployed incrementally by end users and removed as needed, which should be extremely rare. The enforcement mechanism must be effective at reducing unwanted network traffic as it is incrementally deployed. Not all user-administered machines are going to immediately install a mechanism that prevents their machines from being used to attack others; indeed, not all users will want to install such a mechanism. The enforcement mechanism must not rely upon upgraded hardware within the network or widespread deployment and adoption of new protocols. By co-locating the enforcement mechanism with the hosts that they are potentially restricting, our solution can be deployed by individual users without requiring ambitious network hardware or software upgrades.

4 TCP Prototype

We describe a prototype implementation of our solution for TCP. Because TCP is a connection-oriented protocol, we were able to use its existing characteristics to develop a prototype that is virtually invisible to end-hosts. Our enforcement mechanism executes on a separate physical machine whose sole purpose is to act as a gateway between our user-administered host and the larger network. All traffic to and from the host must pass through our enforcement mechanism over the dedicated Ethernet connection.

The system "learns" the host's IP address by observing the source IP address in all outgoing network packets. If the network link goes down or if there is a sustained period of network inactivity, the system re-learns the IP address when outgoing packets are observed. All outgoing packets using a different source IP address are dropped.

We define our network conversation to be the connection established between two (IP:port) pairs using TCP's three-way handshake protocol. We leverage TCP's handshake protocol to determine the start of a network conversation. TCP also contains two distinct techniques for closing a connection: a FIN-ACK sequence initiated by each half of the connection, and a RST packet sent by either side of the connection. As with TCP's connection establishment, we simply leverage this explicit signalling to track the end of a conversation.

As long as neither host is compromised or misbehaving, TCP's built-in control signalling ensures that hosts can terminate any undesired connection. The true merit of our enforcement mechanism is observed when one of the hosts is ignoring the TCP termination messages that it receives.

Imagine that a remote host establishes a TCP connection with our local host, and the local host starts flooding the remote host with network packets. The remote host may send a RST packet to stop the packet flood, but the local host may simply ignore the RST packet and continue to send high rates of unwanted packets. Our enforcement mechanism monitors each established connection to prevent this type of scenario. Once it observes a valid incoming RST packet, the enforcement mechanism drops all outgoing network packets associated with this connection.

In its efforts to restrict unwanted outgoing traffic, the enforcement mechanism must be careful not to allow spoofed RST packets to cause it to incorrectly terminate TCP connections. Additionally, it must prevent a compromised host and a colluding remote attacker from spoofing a connection establishment sequence and using it to attack a third network host. TCP uses the exchange of sequence numbers to provide reasonably good authentication of each endpoint during connection establishment and teardown. This approach, however, relies upon the belief that at least one of the participating hosts is well-behaved and trustworthy. Because both sides of our enforcement mechanism are potentially compromised, it cannot rely upon the validity of TCP's authentication.

Our enforcement mechanism must provide its own endpoint authentication; it does this by adding a random 32-bit nonce to the initial sequence number (ISN) provided by each host during connection establishment. By adding this random value to each host's sequence number, the enforcement mechanism authenticates each endpoint when the modified sequence number is returned. This is the same authentication technique used by standard TCP, but when used by the enforcement mechanism it ensures that two untrusted, colluding hosts cannot subvert the enforcement mechanism using pre-established ISNs. The random nonces are individually generated for each connection establishment seen by the enforcement mechanism; thus the nonce provides the "liveness" property necessary for thwarting replay attacks.

Adding the authenticating nonce to TCP's sequence number requires the enforcement layer to continue modifying all subsequent packets' sequence numbers. It must add the nonce to all outgoing packets' sequence numbers and remove the nonce from all incoming packets' sequence numbers. The enforcement mechanism must also recalculate the checksum for each modified packet; the checksum does not need to be completely recalculated but can simply be updated by the difference between the old and new fields.

The enforcement mechanism maintains per-connection state to track the status of each TCP connection. Our implementation required 108 bytes of connection state for each active connection. Because our mechanism enforces conversation termination for a single host, the number of active connections that we are monitoring should remain small, as will our overall storage requirements.

5 Related Work

A diverse set of techniques and mechanisms have been proposed to address the widespread, damaging nature of modern Internet attacks.

A variety of projects have attempted to characterize network traffic; these characterizations can then be used to filter or identify unwanted network traffic. Network connectivity patterns have been used to characterize both normal and abnormal network traffic ([3], [25], [22], [24]) including the propagation patterns of individual worms ([15], [14], [28].) Studies have quantified denial-of-service [16] activity and spyware [19] seen on the Internet. Worms signatures ([10], [17], [21]) can be used in the identification of traffic containing worms. Other work has focused on the characteristics of "normal" traffic ([12], [27]) for detecting or rate-limiting anomalous behavior.

Our proposed solution is orthogonal to this work in that we require an endhost to determine for itself whether or not a particular stream of network traffic is unwanted. We provide a mechanism whereby the host can request that a packet stream be halted; a host can leverage any of these characterization techniques to decide if the stream is unwanted.

Existing research has proposed introducing new mechanism in the network to identify, account for, and eliminate unwanted traffic. IP Traceback [20] uses network state to identify the path taken by unwanted DoS traffic. Pushback [8] and AITF [2] install packet filters at routers within the network to filter out unwanted traffic. Capability-based networks [1] use packet processing hardware at trust boundaries to enable hosts to communicate while network attacks occur. In the face of congestion, network hardware may selectively mark [6] or drop packets associated with high packet rates.

In contrast with these network-based mechanisms, this work proposes a mechanism that is co-located with a host to prevent unwanted network traffic from being injected into the Internet. We deploy our enforcement mechanisms at potentially malicious traffic sources so that we can drop unwanted traffic before it can impact other hosts. This principle is similarly embraced by network ingress filtering [5], reverse firewalls [13], and IP throttling [26]. Although we allow hosts to push cessation requests upstream like Pushback [8] and AITF [2], we depend upon users' willingness to have mechanism co-located with their hosts to eliminate their need for increased mechanism in the network. By leveraging hardware mechanism at the source of network traffic, our solution can be incrementally deployed at the endhosts. No large-scale network hardware or software upgrades are required before benefits can begin to accrue.

Unlike many source-limiting approaches, our solution does not merely enforce a well-established definition of good behavior, such as limiting the rates of outgoing connections, packets, or source IP addresses. Our work leverages techniques that use feedback mechanisms to indicate when a host is behaving poorly. RED [6] and ECN Nonce [4] use network mechanisms to inform a host that there is network congestion; and TCP uses packet drops to scale back its transmission rate. Our work differs from these approaches in that we only rely upon endhosts to provide negative feedback in the form of requests to terminate malicious conversations.

Finally, a key property of our solution is the independence of our enforcement mechanism. Our enforcement mechanism assumes that it is surrounded by untrustworthy, malicious entities that will try to subvert or disable it. Therefore, the enforcement mechanism must be active in its efforts to prevent malicious traffic. This is in direct contrast to many firewalls [7] and intrusion detection systems [18] which assume that at least one side of the enforcement mechanism is trustworthy.

5.1 Summary

We have argued that we can leverage the good intentions of users to reduce unwanted traffic on the Internet. User-administrated machines are frequently vulnerable to compromise, and once compromised their physical resources can be used to attack other hosts. By co-locating an enforcement mechanism with these well-intentioned hosts, recipients of unwanted traffic can summarily terminate streams of incoming packets from these hosts. If a host has been compromised and is attacking other machines, the victims have the ability to, at least temporarily, terminate the attack.

References

[1] Tom Anderson, Timothy Roscoe, and David Wetherall. Preventing internet denial-of-service with capabilities. *SIGCOMM Comput. Commun. Rev.*, 34(1):39–44, 2004.

[2] Katerina J. Argyraki and David R. Cheriton. Active internet traffic filtering: Real-time response to denial of service attacks. *CoRR*, cs.NI/0309054, 2003.

[3] Daniel R. Ellis, John G. Aiken, Kira S. Attwood, and Scott D. Tenaglia. A behavioral approach to worm detection. In *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 43–53, New York, NY, USA, 2004. ACM Press.

[4] David Ely, Neil Spring, David Wetherall, Stefan Savage, and Tom Anderson. Robust congestion signaling. In *ICNP*, 2001.

[5] Paul Ferguson and Daniel Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing RFC 2267. IETF RFC Publication, January 1998.

[6] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[7] Michael B. Greenwald, Sandeep K. Singhal, Jonathan R. Stone, and David R. Cheriton. Designing an academic firewall: Policy, practice, and experience with surf. In *SNDSS '96: Proceedings of the 1996 Symposium on Network and Distributed System Security (SNDSS '96)*, page 79, Washington, DC, USA, 1996. IEEE Computer Society.

[8] John Ioannidis and Steven M. Bellovin. Implementing pushback: Router-based defense against ddos attacks. In *NDSS*, 2002.

[9] Nick L. Petroni Jr., Timothy Fraser, Jesus Molina, and William A. Arbaugh. Copilot - a coprocessor-based kernel runtime integrity monitor. In *USENIX Security Symposium*, pages 179–194, 2004.

[10] Hyang-Ah Kim and Brad Karp. Autograph: Toward automated, distributed worm signature detection. In *USENIX Security Symposium*, pages 271–286, 2004.

[11] Samuel T. King, Peter M. Chen, Yi-Min Wang, Chad Verbowski, Helen J. Wang, and Jacob R. Lorch. Subvirt: Implementing malware with virtual machines. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy*, page 65, Washington, DC, USA, 1997. IEEE Computer Society.

[12] John McHugh and Carrie Gates. Locality: a new paradigm for thinking about normal behavior and outsider threat. In *NSPW '03: Proceedings of the 2003 workshop on New security paradigms*, pages 3–10, New York, NY, USA, 2003. ACM Press.

[13] Jelena Mirkovic, Gregory Prier, and Peter L. Reiher. Attacking ddos at the source. In *ICNP '02: Proceedings of the 10th IEEE International Conference on Network Protocols*, pages 312–321, Washington, DC, USA, 2002. IEEE Computer Society.

[14] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. The spread of the sapphire/slammer worm. Technical report, CAIDA, ICSI, Silicon Defense, UC Berkeley EECS and UC San Diego CSE, 2003.

[15] D. Moore, C. Shannon, and J. Brown. Code-Red: a case study on the spread and victims of an Internet worm. In *Proc. of Internet Measurement Workshop 2002*, Nov 2002.

[16] David Moore, Geoffrey M. Voelker, and Stefan Savage. Inferring Internet denial-of-service activity. In *USENIX Security Symposium*, 2001.

[17] James Newsome, Brad Karp, and Dawn Song. Polygraph: Automatically generating signatures for polymorphic worms. In *IEEE Symposium on Security and Privacy*, 2005.

[18] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(23–24):2435–2463, 1999.

[19] Stefan Saroiu, Steven D. Gribble, and Henry M. Levy. Measurement and analysis of spyware in a university environment. In *NSDI*, pages 141–153, 2004.

[20] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for ip traceback. In *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 295–306, New York, NY, USA, 2000. ACM Press.

[21] Sumeet Singh, Cristian Estan, George Varghese, and Stefan Savage. Automated worm fingerprinting. In *OSDI*, pages 45–60, 2004.

[22] S. Staniford-Chen et al. GrIDS—A graph based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, volume 1, pages 361–370, October 1996.

[23] Symantec. Symantec internet security threat report, volume ix, Mar 2006.

[24] Godfrey Tan, Massimiliano Poletto, John Guttag, and Frans Kaashoek. Role Classification of Hosts within Enterprise Networks Based on Connection Patterns. In *The USENIX Annual Technical Conference 2003*, San Antonio, TX, June 2003.

[25] T. Toth and C. Kruegel. Connection-history based anomaly detection, 2002.

[26] J. Twycross and M.M. Williamson. Implementing and testing a virus throttle. In *USENIX Security Symposium*, 2003.

[27] Matthew M. Williamson. Throttling viruses: Restricting propagation to defeat malicious mobile code. In *ACSAC '02: Proceedings of the 18th Annual Computer Security Applications Conference*, page 61, Washington, DC, USA, 2002. IEEE Computer Society.

[28] Cynthia Wong, Stan Bielski, Jonathan M. McCune, and Chenxi Wang. A study of mass-mailing worms. In *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 1–10, New York, NY, USA, 2004. ACM Press.