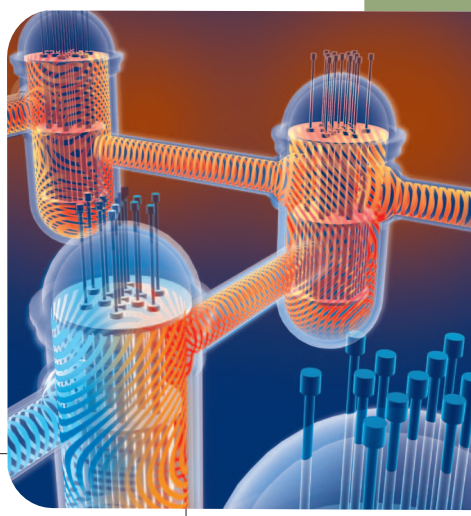
Collaborative Internet Worm Containment



Large-scale worm outbreaks that lead to distributed denial-of-service attacks pose a major threat to Internet infrastructure security. Fast worm containment is crucial for minimizing damage and preventing flooding attacks against network hosts.

Min Cai, Kai Hwang, Yu-Kwong Kwok, Shanshan Song, and Yu Chen
University of Southern California

Recent Internet worm outbreaks have infected hundreds of thousands of Internet servers and user machines within minutes,[1] causing billions of dollars in losses for businesses, governments, and service providers.[2] CodeRed, for example, affected more than 359,000 Web servers in 14 hours; Slammer achieved its maximum Internet-wide scanning rate (55 million scans per second) in a few minutes. The high stakes involved have inspired numerous research projects, through which industry and academic institutions are working to strengthen local-area and wide-area networked systems' abilities to fend off cyberattacks.

To that end, we propose deploying fast, scalable security overlay networks based on distributed hash tables (DHTs)[3] to facilitate high-speed intrusion detection and alert-information exchange. Fortifying the Internet infrastructure with such a solution could benefit many security-sensitive applications, such as digital government, critical infrastructures, grid computing, e-commerce, and law enforcement. The broader impacts are far reaching in science, education, business, and homeland security.

An effective system for worm detection and cyberspace defense must have the following capabilities:

- robustness and resilience in performing security functions in real-life Internet environments;[4,5]
- cooperation among multiple sites, enabled by trust integration and alert-correlation methodologies;[6]
- responsiveness to unexpected worm or flooding attacks, enabled by fast anomaly detection and distributed denial-of-service (DDoS) defense;[7] and
- efficiency and scalability, enabled by fast worm-signature detection and dissemination,[8] as well as accurate traffic monitoring for tracking DDoS attack-transit routers.[9]

With these attack-resilience goals in mind, we are developing a prototype NetShield cyberspace defense system[6] for containing the spread of worms and defending against DDoS flooding attacks (http://gridsec.usc.edu/). Here, we focus on two major components in the NetShield system: the WormShield system for detecting and disseminating worm signatures and a traffic-monitoring scheme for tracking DDoS flooding attacks. We present experimental results for these two component systems derived from large-scale simulations.

DHT-based security overlay networks

Many researchers have suggested the use of overlay networks for security services, building on collaborative nodes to support fast alert correlation and group communications.[4,5] Network-layer overlays built with virtual links by IP tunneling are generally faster for message forwarding than popular application-layer overlays, such as peer-to-peer (P2P) networks. However, introducing new functionality into network-layer overlays requires modifications to OS kernels and other infrastructure features, which makes them impractical for ordinary users. To address this fact, we've explored the use of distributed P2P networks with DHT systems for fast, resilient look-up services.

The idea is to maintain structured overlay networks among participating peers and use simple message routing rather than flooding. The basic functionality that DHTs provide is in the `lookup(key)` operation, which returns the identity of the node in which the ob-

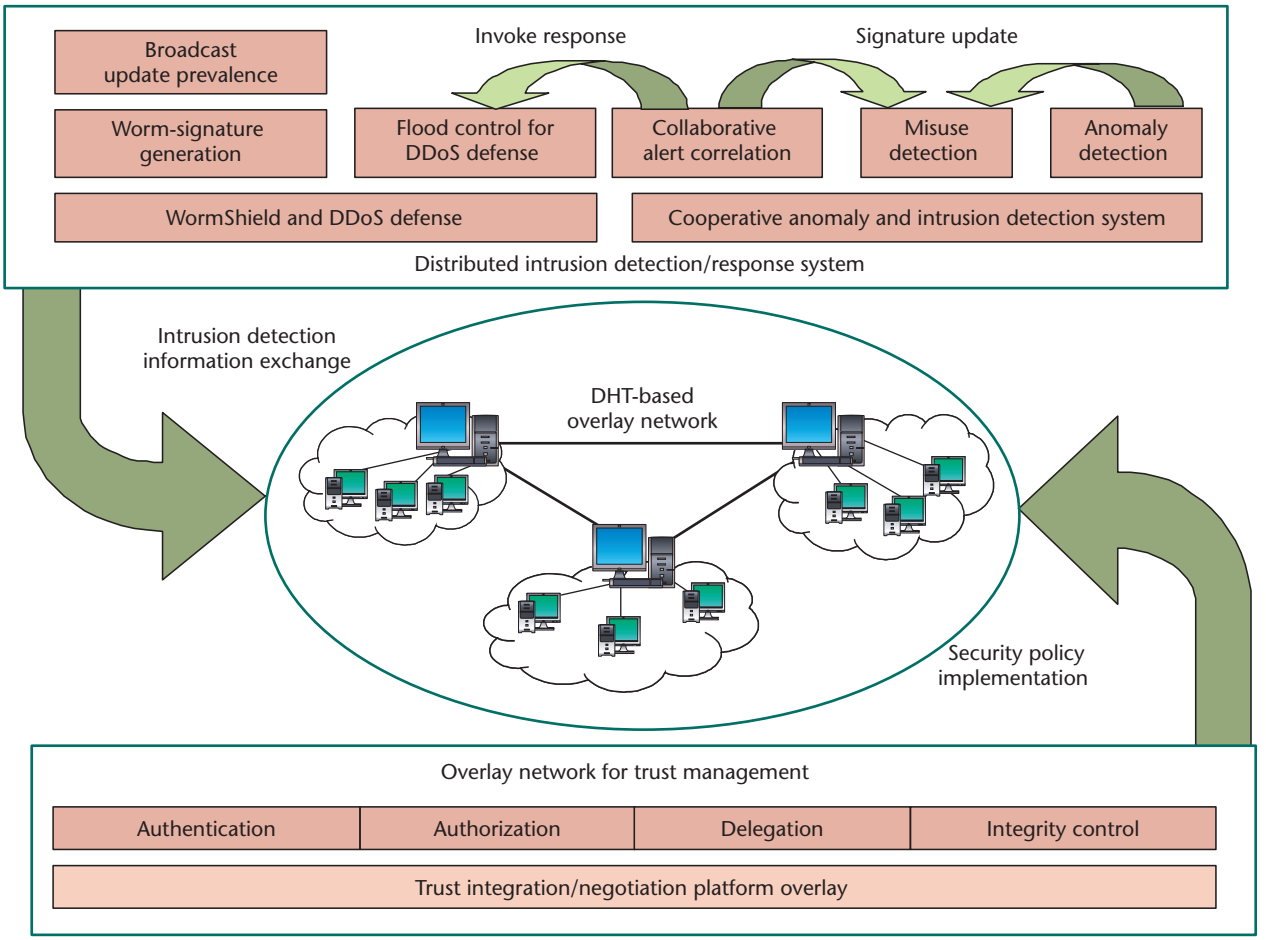


Figure 1. The NetShield system architecture and trust integration over a distributed hash table (DHT) overlay. The system performs trust management across multiple administrative domains, suppresses Internet worm outbreaks, and defends against distributed denial-of-service (DDoS) flooding attacks.

ject with the specified key is stored. When a node issues a `lookup(key)` request, the system routes the lookup message to the node responsible for the key. DHT systems can guarantee to finish a lookup operation in $O(\log N)$ hops for a network with N nodes. This implies that the DHT overlay is scalable to very large networks.

Figure 1 illustrates the NetShield architecture, which is built on a DHT-based overlay network that integrates multiple security services including attack/intrusion monitoring, detection and defense, alert correlation, and security-update delivery. Our design is based on the Chord system,[3] which emphasizes efficient routing, reliability, scalability, robustness to failure, and self-organization. We extend the DHT overlay to withstand DDoS, routing, and storage-and-retrieval attacks.

Our approach differs from the Domino[10] and Security Overlay Service (SOS)[4] systems in that we propose an integrated architecture for various security services at the P2P level. We aim to reduce the communication overhead by using DHT `lookup(key)` functionality. NetShield also supports encrypted tunneling over multiple domains, multicast-based authentication, dissemination of newly detected worm signatures, and security binding in Internet services.

Internet worm detection and containment

Recent seminal work suggests that automatically generating worm signatures by analyzing payload contents and address dispersion offers promising results (see the "Related work in worm containment" sidebar on p. 32 for an overview). However, most scanning worms are dispersed over the entire Internet when they start to spread. In the early stages, it's difficult to observe significant anomalies and accumulate enough payload contents with worms at individual edge networks. Instead, we need to correlate information from multiple edge networks to detect worm signatures quickly and accurately—especially for flash worms.[11]

To face this challenge, we developed the WormShield

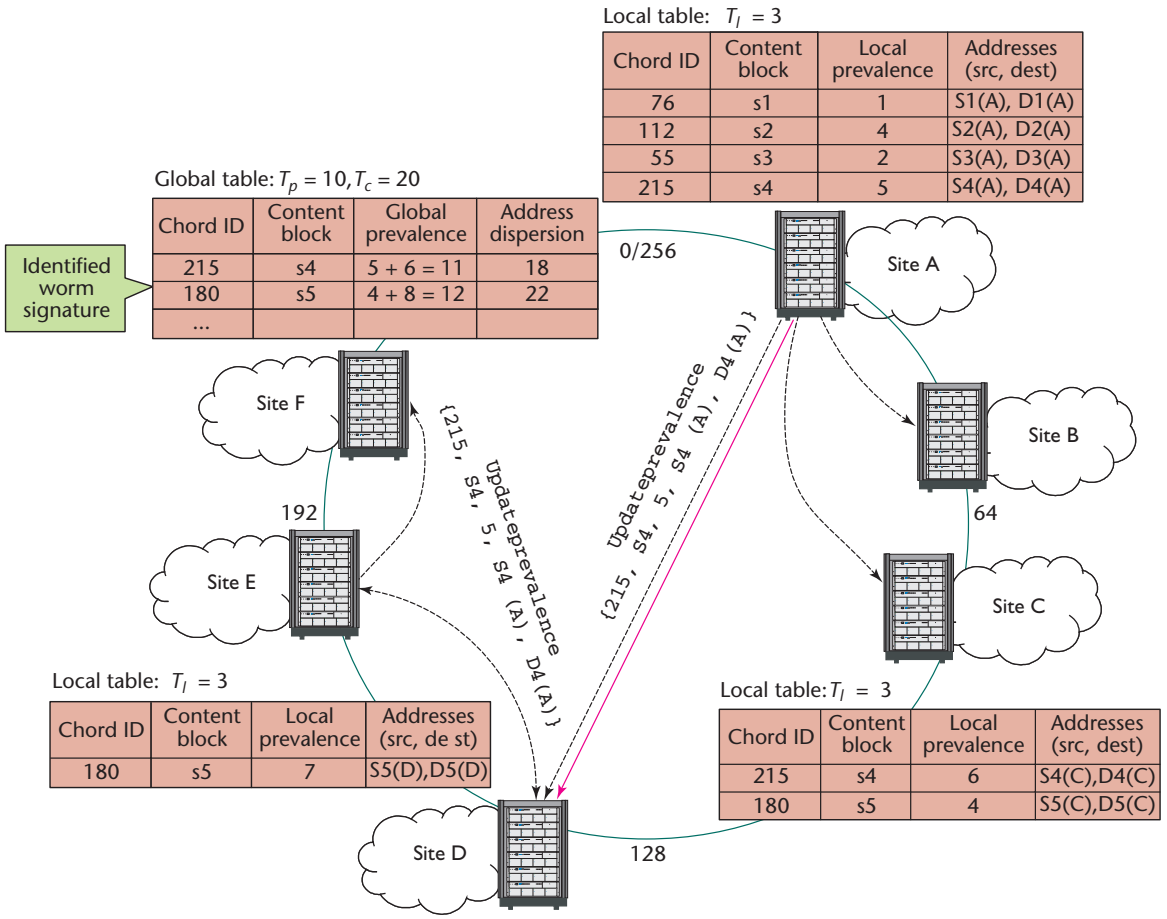


Figure 2. The WormShield architecture. In this example, six worm-monitoring sites are deployed in six edge networks. This DHT-based overlay system performs distributed worm monitoring, anomaly detection, signature updating, alert correlation, and automated intrusion response.

worm-signature detection and characterization system, which is designed to identify and contain unknown worms before they infect most vulnerable hosts. The system includes a set of geographically distributed monitors located in multiple administrative domains.[8] These monitors are self-organized into a structured P2P overlay network based on the Chord algorithm.[3] Each monitor is deployed on the edge network's demilitarized zone (DMZ), where it analyzes all incoming packets passing through it. Figure 2 shows an example WormShield network with six sites deployed.

Signature generation and dissemination

Figure 3 illustrates the worm-signature detection and dissemination process. Each WormShield monitor i uses the Rabin footprint algorithm[12] to compute the content blocks in packet payloads. It then updates the local prevalence table for each content block j, denoted by $L(i, j)$, which tracks the number of occurrences of content block j observed by monitor i. It also remembers the set of source addresses $S(i, j)$ and destination addresses $D(i, j)$ corresponding to content block j. Once $L(i, j)$ is greater than a local prevalence threshold T_l and the address dispersion $|S(i, j)| + |D(i, j)|$ is greater than a local address-dispersion threshold T_s, monitor i starts to update the global prevalence $P(i, j)$ as well as the global address dispersion $C(j)$ for content block j.

For each content block, we select a root monitor using the consistent hashing scheme in Chord, which works by mapping arbitrary content blocks to the monitors in a load-balanced manner.[3] The root monitor maintains its corresponding content block's aggregated prevalence and address dispersion. When $P(j)$ is greater than a global prevalence threshold T_p, and $C(j)$ is greater than an address-dispersion threshold T_c, the root monitor identifies the content block j as a potential worm signature. It then constructs a multicast tree on the overlay network and disseminates the signature to all monitors in the WormShield network.

Each monitor could automatically deploy the received worm signature in local signature-based intrusion detection systems, such as Snort (www.snort.org) or Bro

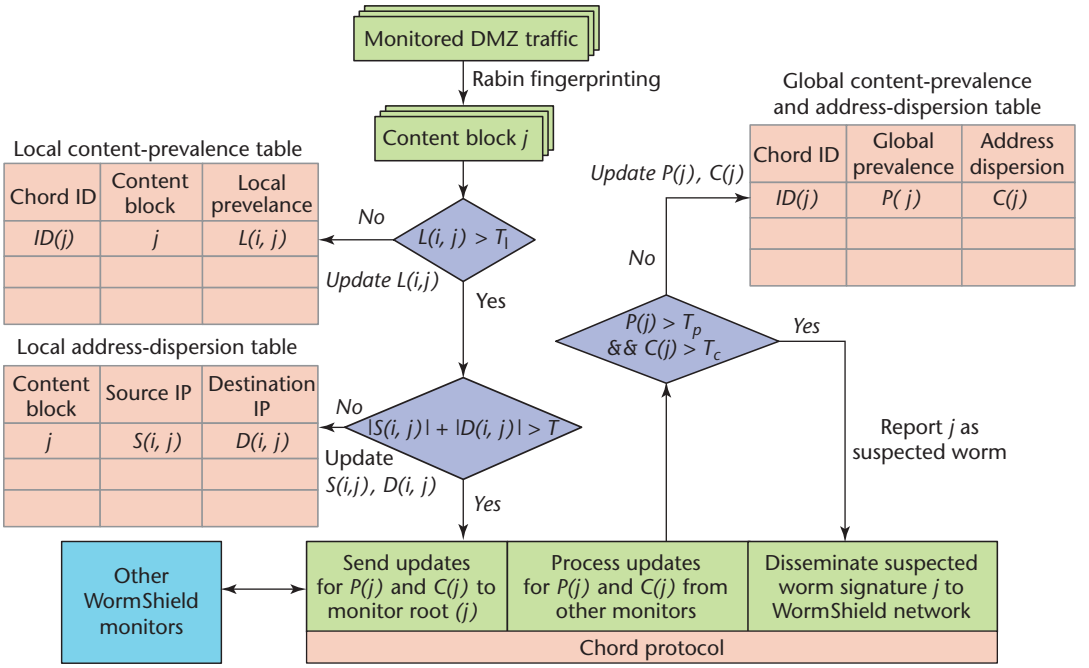


Figure 3. The worm-signature detection and dissemination process. Each WormShield monitor carries out three key mechanisms: local prevalence with address dispersion, global prevalence with address dispersion, and dissemination of suspected worm signatures.

(http://bro-ids.org), or define different policies for importing signatures generated by other root monitors. For example, they might notify local security administrators before activating signature filtering or rate limiting.

Simulated worm-spreading experiments

We evaluated WormShield with a large-scale simulation using a method similar to that employed by David Moore and colleagues[7] and Hyang-Ah Kim and Brad Karp.[13] Specifically, we simulated two variants of CodeRed worms on a real Internet configuration of 105,246 edge networks in 11,342 autonomous systems (ASes) with 338,652 vulnerable hosts. We assigned IP address ranges for edge networks and ASes based on a Border Gateway Protocol (BGP) table snapshot from the University of Oregon's RouteViews project (www.routeviews.org) for the date of the CodeRed outbreak (19 July 2001).

We truncated all address blocks that were larger than class B networks (/16 network, so-called because its network prefix is the first 16 bits) because it's unreasonable to attempt to monitor class A networks (/8 network) with only one monitor. We also ignored address blocks smaller than class C networks (/24 network) because those address blocks often represent router link interfaces rather than potentially vulnerable end hosts. In the end, we identified 6,378 vulnerable ASes and 61,216 vulnerable edge networks (containing at least one vulnerable host). Like Kim and Karp, we assumed that 50 percent of the address space within the vulnerable ASes was populated with reachable hosts, and that 25 percent of those hosts were running Web servers. Given that CodeRed worms send infecting packets only after establishing TCP connections with targets, only 1/8 of the address space within those vulnerable ASes would receive TCP payloads with infecting packets.

We simulated a CodeRedI-v2-like worm, which uniformly probes the entire IP address space except for 244.0.0.0/8 (multicast) and 127.0.0.0/8 (loopback), as these are invalid addresses for any end host. We also simulated a CodeRedII-like worm, which probes a completely random IP address 1/8 of the time, addresses in the same class A network half the time, and addresses in the same class B network 3/8 of the time. Both worms sent 10 probes per second and began the simulation with 25 infected hosts.

Figure 4a shows the infection progress for these two simulated worms. The results for the CodeRedI-v2 worm are very similar to Moore and colleagues' simulation. Because CodeRedII uses subnet-preferred scanning, which is biased toward probing hosts in local subnets rather than uniform scanning, it spreads much faster—CodeRedII infected 50 percent of the vulnerable population (169,326 hosts) in 66 minutes, whereas CodeRedI-v2 took 219 minutes to infect the same number of hosts.

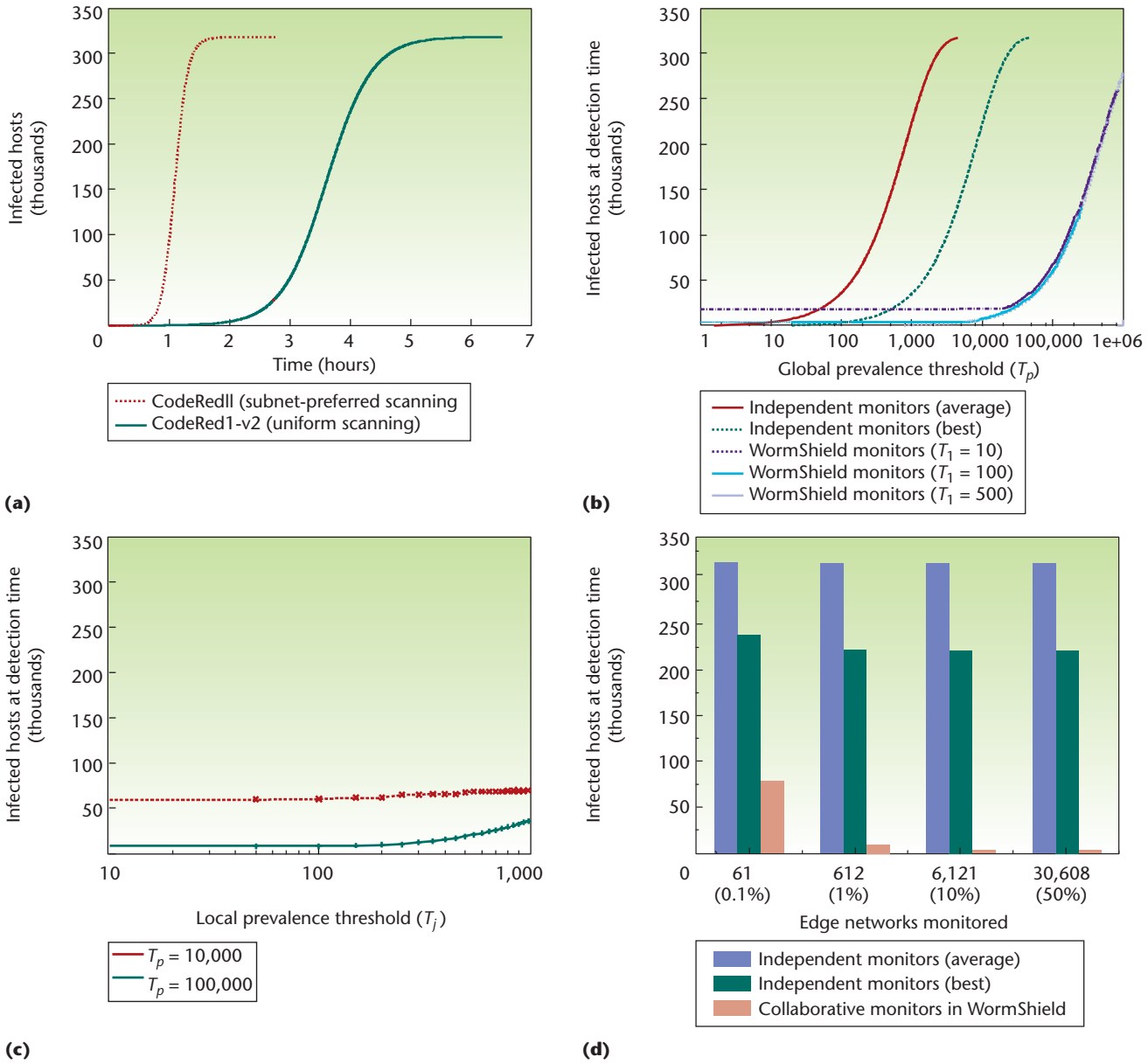


Figure 4. Signature detection and growth of infected hosts for simulated CodeRed worms. The simulations ran on an Internet configuration of 105,246 edge networks in 11,342 autonomous systems containing 338,652 vulnerable hosts. (a) Infection progress for two CodeRed variants shows that worms with subnet-preferred scanning spread faster than those with uniform scanning. (b) The effects of the global prevalence threshold show that collaborative monitors can detect signatures much faster than independent monitors. (c) The effects of the local prevalence threshold show that signature detection slows down slightly as local prevalence threshold increases. (d) The effects of monitors deployed over different edge networks show that WormShield monitors can reduce more infected hosts at detection time than independent monitors when edge networks being monitored increase from 0.1 percent to 50 percent of total networks monitored.

Figure 4b shows the number of vulnerable hosts infected when WormShield succeeds in detecting the worm signature. We plot the results as a function of the global prevalence threshold required to trigger signature detection. In this experiment, we monitored 1 percent of the vulnerable edge networks (612 out of 61,216 networks) with both independent monitors and collaborative WormShield monitors.

When the global prevalence threshold was 1,000—often the number required in distinguishing innocent

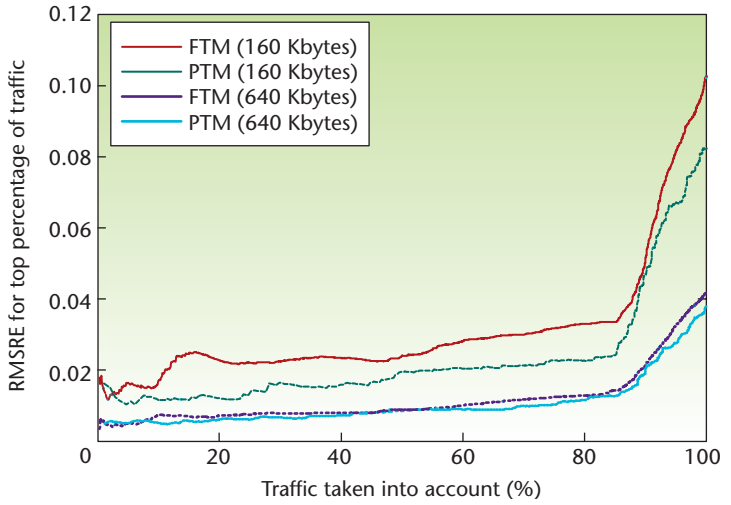


Figure 5. Root-mean-squared relative error (RMSRE) of packet-level (PTM) and flow-level traffic matrix (FTM) elements for various percentages of traffic. It is generally easier to accurately estimate larger TM elements than smaller ones; accuracy improves significantly for PTM and FTM as the top percentage of traffic taken into account decreases.

content blocks—the worm infected 223,510 hosts before the average independent monitors succeeded in detecting its signature. Even the best independent monitors detected worm signatures only after 35,219 hosts were infected. WormShield monitors, on the other hand, detected the signatures by the time 1,354, 8,127, and 18,287 hosts were infected (for local prevalence thresholds of 10, 100, and 500, respectively).

Figure 4c plots the effect of the local-prevalence threshold required to trigger global-prevalence updates. With a global threshold of 100,000, the number of infected hosts at detection time increased only slightly when the local threshold increased from 10 to 1,000. For a global threshold of 10,000, however, the number of infected hosts at detection time increased from 9,482 to 35,897 when the local threshold increased from 200 to 1000. This suggests that the local threshold must be set at a value smaller than 200 for a global threshold of 10,000.

We also compared WormShield's signature-detection speeds when different fractions of the vulnerable edge networks were monitored. Figure 4d shows the number of vulnerable hosts infected at detection time when 61, 612, 6,121 and 30,608 edge networks were monitored. With independent monitors, worm-detection speeds didn't improve as we increased the number of monitors. Because WormShield monitors work collaboratively, however, total infected hosts at detection time decreased from 77,889 to 4,063 as we increased the number of monitored edge networks.

Fast and accurate traffic monitoring

Spreading worms often trigger DDoS flooding attacks. Effective containment of worms can significantly reduce the number of infected hosts, as well as stem the creation of "zombie" end-user machines that can be used later as attackers. However, we can't completely prevent DDoS attacks by simply containing worm outbreaks. Backbone ISPs often need to monitor traffic volumes among routers to identify attack-transit routers (ATRs)—ingress routers that unwittingly forward malicious DDoS attack flows.

The network traffic matrix (TM), which represents the amount of traffic in bytes, packets, or flows between ingress and egress routers, is very useful for identifying ATRs. We consider packet-level TM (PTM) and flow-level TM (FTM). We proposed a cardinality-based TM-measurement approach (CBTM) for both PTM and FTM. We denote $X(t_k)$ as the TM for time slot t_k. Let $S_i^+(t_k)$ be the set of traffic entering the network from ingress router r_i during t_k and $S_j^-(t_k)$ be the set of traffic leaving from egress router r_j during t_k. According to the definition of TM, we have

$$X_{i,j}(t_k) = |S_i^+(t_k) \cap S_j^-(t_k)| = |S_i^+(t_k)| + |S_j^-(t_k)| - |S_i^+(t_k) \cup S_j^-(t_k)|.$$

Obviously, it's impractical to collect all original traffic from routers and perform the union operations. Instead, we compress traffic sets $S_i^+(t_k)$ and $S_j^-(t_k)$ into very small *cardinality summaries* at routers r_i and r_j, denoted by $C_i^+(t_k)$ and $C_j^-(t_k)$. We estimate $|S_i^+(t_k)|$ and $|S_j^-(t_k)|$ from $C_j^+(t_k)$ and $C_j^-(t_k)$, respectively, using our proposed adaptive cardinality counting algorithm,[9] which extends the LogLog counting algorithm designed by Marianne Durand and Philippe Flajolet.[14] By *max-merging* $C_i^+(t_k)$ and $C_j^-(t_k)$, we can also estimate $|S_i^+(t_k) \cup S_j^-(t_k)|$ from the merged cardinality summary.

For PTM, we use the invariant IP header fields and first few bytes of payload as a packet identity. We then compress packet traffic data into cardinality summaries using their identities. For FTM, we compress IP flows at each router using five-tuple flow identities consisting of source IP addresses and port numbers, destination IP addresses and port numbers, and protocol identifiers. In CBTM, each cardinality summary takes only $O(\log \log N)$ storage capacity to record the traffic set of N packets or flows.[9] For example, we need only 640 Kbytes of storage to record traffic data at 40 Gbits-per-second line speed.

Figure 5 illustrates our trace-driven simulation results on the accuracy of PTM and FTM monitoring. To better understand the error distribution among TM elements, we use the root-mean-squared relative error (RMSRE) metrics.[15] For a given percentage of traffic, RMSRE reflects the average relative error of the largest TM elements whose traffic sum is that percentage of the total traffic in

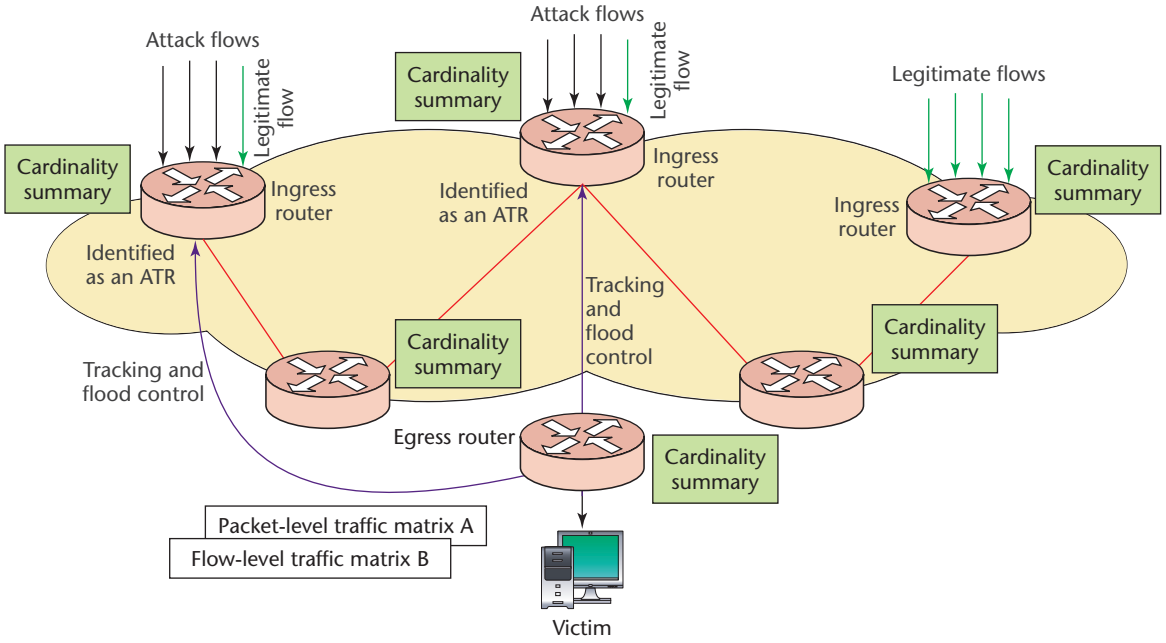


Figure 6. Traffic matrix monitoring for tracking attack-transit routers (ATRs). Collaborative routers can perform distributed tracking of ATRs by correlating the packet-level (PTM) and flow-level traffic matrix (FTM). Here, the egress router identifies two potential ATRs by correlating the PTM and FTM.

all elements. Figure 5 shows that when all traffic is taken into account, the RMSRE of PTM and FTM with cardinality summaries of 640 Kbytes are 0.038 and 0.042, respectively. However, when we consider only 90 percent of the traffic in the largest elements, both decrease to 0.021 and 0.022, respectively. Similar trends hold for estimations with cardinality summaries of 160 Kbytes.

Monitoring the spatial and temporal behaviors of PTM and FTM can help ISP operators track DDoS flooding attacks. During a flooding attack, the victim often receives a large surge of small flows because flooding packets usually have randomly spoofed source IP addresses. Because they don't use sampling on packets, our FTMs can capture all these small flows. The egress router to the victim can identify the ATRs (as illustrated in Figure 6) by correlating the PTM and FTM. The ATRs often show significant increases in the number of flows but no corresponding increase in the number of packets to the victim. For increased packets $\Delta P_{i,j}(t_k)$ and flows $\Delta F_{i,j}(t_k)$ from ingress router r_i to the victim's egress router r_j in time slot t_k, we can estimate the average size of new flows $z_i(t_k)$ at router r_i as $z_i(t_k) = \Delta P_{i,j}(t_k)/\Delta F_{i,j}(t_k)$. Therefore, if an ingress router has exceptionally large $\Delta F_{i,j}(t_k)$ and small $z_i(t_k)$, we can identify it as a suspected ATR. Anukool Lakhina and colleagues also use the time serials of PTM and FTM to diagnose various network anomalies, including DDoS attacks.[16] Our fast PTM and FTM monitoring can provide very useful online inputs for Lakhina's algorithm to diagnose DDoS flooding attacks.

Our large-scale simulation showed that deploying collaborative WormShield monitors on just 1 percent of the vulnerable edge networks can let us detect worm signatures roughly 10 times faster than with independent monitors. To handle the major threats posed today by software vulnerabilities and naïve users, we plan to develop a trust-negotiation layer on top of the P2P overlay network to coordinate distributed detection and defense activities. This subsystem will identify and isolate compromised peers to help minimize negative effects. Providing the means to isolate compromised peers could also let ISPs and other administrative entities share attack data and request response actions from entities that remain trusted.

In swiftly dealing with worm outbreaks, the major research challenge is still in containment. In particular, we need automated signature generation and fast suppression of malicious flows. We're in the process of implementing the WormShield system and will evaluate its performance with real worm-spreading experiments on the Cyber Defense Technology Experimental Research (Deter) test bed (www.isi.edu/deter/). We will also evaluate our PTM and FTM correlation schemes to diagnose network-wide traffic anomalies caused by DDoS flooding attacks in a real-life environment. □

Related work in worm containment

Several university and industrial projects are working to develop worm-containment systems. However, as most are in the initial research-and-development stages, widely deployed antiworm software products have yet to emerge from these efforts.

The Earlybird system (www.cs.ucsd.edu/groups/sysnet/), by Sumeet Singh and colleagues,[1] and Autograph (http://www-2.cs.cmu.edu/~hakim/autograph/), by Hyang-Ah Kim and Brad Karp,[2] were the first to automatically generate worm signatures by analyzing packet payload-content prevalence and address dispersion.

Earlybird uses a content-sifting approach to detect content prevalence and scaled bitmaps to estimate address dispersion. Sensors sift through traffic on configurable address space zones and report signatures to an aggregator, which coordinates real-time updates from the sensors, merges related signatures, and activates network- or host-level blocking services. Earlybird's key contribution is its clever algorithmic design, which supports a robust, scalable wire-speed implementation in a single worm sensor. On the other hand, the system supports distribution only through the centralized aggregator. It doesn't support information sharing among different sensors in content prevalence. Without such collaborative features, we expect that Earlybird's performance will match the best independent monitors in our simulation.

Autograph automatically generates signatures from worms propagating with TCP transport. The system analyzes the prevalence of partial-flow payloads and produces signatures that exhibit high true positives and low false positives. Autograph has better support for distributed deployment; it uses application-level multicast to share port-scan reports among distributed monitors. As in Earlybird, however, the monitors don't share worm payload information, so each accumulates only as much payload as an independent monitor in our simulation.

Nicholas Weaver and colleagues developed a fast scan-detection and suppression algorithm based on the Threshold Random Walk online malicious-host-detection algorithm. The simplifications in their algorithm make it suitable for both hardware and software implementation (www.icsi.berkeley.edu/Networks/).[3] Their project also enhances containment through collaboration among containment devices.

Cliff Zou and colleagues proposed a worm-monitoring and early warning system, called trend detection (http://tennis.ecs.umass.edu/~czou/research.htm).[4] Based on some worm-propagation dynamic models, the trend detection system detects the presence of a worm in its early stage by using a Kalman filter estimation algorithm.

Our NetShield[5] system (http://gridsec.usc.edu/) uses a distributed hash table-based overlay system for fast worm detection with automated signature generation and dissemination. We also aim at monitoring network traffic matrix at backbone ISPs to track network anomalies caused by distributed denial-of-service (DDoS) flooding attacks.

Rather than analyzing network traffic, the Columbia Worm Vaccine project[6] and the Microsoft Shield system[7] adopt end-system approaches to prevent vulnerable hosts from being infected.

Acknowledgments

The US National Science Foundation supports this work under ITR grant number ACI-0325409. We thank the GridSec research group at the University of Southern California for helpful discussions that improved this article.

References

1. S. Savage, "Internet Outbreaks: Epidemiology and Defenses," Keynote Address, Internet Society Symp. Network and Distributed System Security (NDSS 05), 2005; www.cs.ucsd.edu/~savage/papers/InternetOutbreak.NDSS05.pdf.
2. A. Chakrabarti and G. Manimaran, "Internet Infrastructure Security: A Taxonomy," *IEEE Network*, Nov. 2002, pp. 13–21.
3. I. Stoica et al., "A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Trans. Networking*, vol. 11, no. 1, 2003, pp. 17–32.
4. A.D. Keromytis, V. Misra, and D. Rubenstein, "SOS: Secure Overlay Services," *Proc. ACM SIGCOMM*, ACM Press, 2002, pp. 61–72.
5. J. Li, P. Reiher, and G. Popek, "Resilient Self-Organizing Overlay Networks for Security Update Delivery," *IEEE J. Selected Areas in Comm.*, Jan. 2004, pp. 189–202.
6. K. Hwang, Y. Chen, and H. Liu, "Defending Distributed Systems against Malicious Intrusions and Network Anomalies," *Proc. IEEE Workshop Security in Systems and Networks* (SSN 05), IEEE CS Press, 2005, pp. 286–286.
7. D. Moore et al., "Internet Quarantine: Requirements for Containing Self-Propagating Codes," *Proc. IEEE INFOCOM*, IEEE CS Press, 2003, pp. 1901–1910.
8. M. Cai et al., "WormShield: Collaborative Worm Signature Detection Using Distributed Aggregation Trees," tech. report TR 2005-10, Internet and Grid Computing Lab, Univ. of Southern California, 2005; http://gridsec.usc.edu/TR/TR-2005-10.pdf.
9. M. Cai et al., "Fast and Accurate Traffic Matrix Measurement Using Adaptive Cardinality Counting," tech. report TR 2005-12, Internet and Grid Computing Lab, Univ. of Southern California, 2005; http://gridsec.usc.edu/TR/TR-2005-12.pdf.
10. V. Yegneswaran, P. Barford, and S. Jha, "Global Intrusion Detection in the Domino Overlay System," *Proc. Network and Distributed System Security* (NDSS 04), Internet Soc., 2004, pp. 79–95.
11. S. Staniford et al., "The Top Speed of Flash Worms," *Proc. Workshop on Rapid Malcode* (WORM 04), ACM Press, 2004, pp. 33–42.

Columbia's Network Worm Vaccine (http://nsl.cs.columbia.edu/projects/wormv/) employs a collection of sensors that detects and captures potential worm-infection vectors based on a set of heuristics. Worm Vaccine uses source code transformations to quickly apply patches to vulnerable segments of targeted applications and then tests the patched applications' resistance to the infection vectors.

Microsoft Research's Shield project (http://research.microsoft.com/research/shield/) installs vulnerability-specific and exploit-generic network filters in end systems once a vulnerability is discovered and before a patch is applied. These filters examine traffic coming in or out of the applications, and drop or correct traffic that exploits vulnerabilities. The system is resilient to polymorphic or metamorphic variations of exploits.

On the research front, Symantec is currently working on a solution to support multiple platforms and protect all network tiers (www.icir.org/vern/worm04/carey.ppt). The company provides patching based on vulnerability information, real-time backup, early-warning and monitoring systems, proactive host and network blocking, and reactive technologies.

Large-scale worm-containment experiments are very much needed. The Cyber Defense Technology Experimental Research (Deter) test bed provides a good platform for performing worm-control experiments (www.isi.edu/deter/).[8] To consolidate the effort, the US National Science Foundation has just funded a new Center for Internet Epidemiology and Defense at the University of California, San Diego, in collaboration with the Berkeley International Computer Science Institute. The annual *ACM Workshop on Rapid Malcode* (WORM) is a good source of information on recent research in this field (WORM 2004 information is available at www.icir.org/vern/worm04/worm04-program.html).

References

1. S. Singh et al., "Automated Worm Fingerprinting," *Proc. Usenix Symp. Operating System Design and Implementation*, Usenix Assoc., 2004, pp. 45–60.
2. H.A. Kim and B. Karp, "Autograph: Toward Automated Distributed Worm Signature Detection," *Proc. Usenix Security Symp.*, Usenix Assoc., 2004, pp. 271–286.
3. N. Weaver, S. Staniford, and V. Paxson, "Very Fast Containment of Scanning Worms," *Proc. 13th Usenix Security Symp.*, Usenix Assoc., 2004, pp. 29–44.
4. C.C. Zou et al., "Monitoring and Early Warning for Internet Worms," *Proc. 10th ACM Conf. Computer and Comm. Security* (CCS 03), ACM Press., 2003, pp. 190–199.
5. K. Hwang et al., "GridSec: Trusted Grid Computing with Security Binding and Self-Defense against Network Worms and DDoS Attacks," *Int'l Workshop on Grid Computing Security and Resource Management* (GSRM 05), Springer-Verlag, 2005.
6. S. Sidiroglou and A.D. Keromytis. "Countering Network Worms through Automatic Patch Generation," to appear in *IEEE Security & Privacy*, 2005.
7. H.J. Wang et al., "Shield: Vulnerability-Driven Network Filters for Preventing Known Vulnerability Exploits," *Proc. ACM SIGCOMM*, ACM Press, 2004.
8. R. Bajcsy et al., "Cyber Defense Technology Networking and Evaluation," *Comm. ACM*, vol. 47, no. 3, 2004, pp. 58–61.

12. M.O. Rabin, "Fingerprinting by Random Polynomials," tech. report 15-81, Center for Research in Computing Technology, Harvard Univ., 1981.
13. H.A. Kim and B. Karp, "Autograph: Toward Automated Distributed Worm Signature Detection," *Proc. Usenix Security Symp.*, Usenix Assoc., 2004, pp. 271–286.
14. M. Durand and P. Flajolet, "LogLog Counting of Large Cardinalities," *Proc. European Symp. Algorithms*, LNCS 2832, Springer-Verlag, 2003, pp. 605–617.
15. Y. Zhang et al., "Fast Accurate Computation of Large-scale IP Traffic Matrices from Link Loads," *Proc. SIGCOMM*, ACM Press, 2003, pp. 301–312.
16. A. Lakhina, M. Crovella, and C. Diot, "Characterization of Network-Wide Anomalies in Traffic Flows," *Proc. ACM/SIGCOMM Internet Measurement Conf.*, ACM Press, 2004, pp. 201–206.

Min Cai *is pursuing a PhD in computer science at the University of Southern California. His research interests include distributed worm-detection and DDoS-defense systems, peer-to-peer, grid computing, and Semantic Web technologies. Cai received a BS and an MS in computer science from Southeast University, China. Contact him at mincai@usc.edu.*

Kai Hwang *is a professor and director of the Internet and Grid Computing Laboratory at USC. An IEEE fellow, he specializes in computer architecture, parallel processing, Internet security, and distributed computing systems. He also leads the GridSec project at USC. Hwang received a PhD in electrical engineering and computer science from the University of California, Berkeley. Contact him at kaihwang@usc.edu.*

Yu-Kwong Kwok *is an associate professor of electrical and electronic engineering at the University of Hong Kong (HKU), currently serving as a visiting associate professor at USC. His research interests include grid and mobile computing, wireless communications, and network protocols. Kwok received a PhD in computer science from the Hong Kong University of Science and Technology (HKUST). Contact him at ykwok@hku.hk.*

Shanshan Song *is a PhD student in the computer science department at USC. Her research interests include trust management in grid and P2P systems, and security-driven scheduling algorithms for computational grids. Song received a BS in computer science from the University of Science and Technology of China. Contact her at shanshan.song@usc.edu.*

Yu Chen *is a PhD student in the electrical engineering department at USC. His research interests include Internet security, DDoS attack detection and defense, Internet traffic analysis, and distributed security infrastructure. Chen received a BS in opto-electronic engineering from Chongqing University, China. Contact him at cheny@usc.edu.*