LEET '08: First USENIX Workshop on Large-Scale Exploits and Emergent Threats

San Francisco, CA
April 15, 2008

ATTACKER BEHAVIOR

Summarized by Joshua Mason (josh@jhu.edu)

- ***On the Spam Campaign Trail***

Christian Kreibich, International Computer Science Institute; Chris Kanich, Kirill Levchenko, Brandon Enright, and Geoffrey M. Voelker, University of California, San Diego; Vern Paxson, International Computer Science Institute; Stefan Savage, University of California, San Diego

Christian Kreibich presented data he and his collaborators gathered about the Storm botnet. The data was collected by first reverse engineering and subsequently infiltrating the botnet with the intention of discerning email address harvesting properties, spam delivery efficacy, and the size of individual spam campaigns. Data capture was accomplished by running 16 virtual machines infected by Storm and situating the nodes at several levels in the Storm hierarchy while disallowing malicious activity such as actually sending spam.

Running live instances of the Storm botnet led to several interesting discoveries. First, Christian discussed the spam templating functionality, which allows spammers to craft messages using a variety of macros. These macros can then be substituted with random data to make emails containing the same general message difficult to cluster. They observed 14 different macros used during their deployment and discovered 10 more with experimentation. The team also discovered dictionaries for use in macro values (e.g., subject lines and domain names) and various lists of email addresses (hit lists) used in different spam campaigns.

Kreibuch went on to give a myriad of different statistics on the spam traffic they observed. They saw over 100,000 command and control connections for worker nodes of the Storm network and were able to collect 172,000 spam templates. They also observed 272,546 harvest reports that contained information gathered from worker nodes. Perhaps the most staggering statistic was the number of targeted email addresses, coming in at 66.7 million. A survey of these addresses revealed some fairly comical addresses such as "first.lady@whitehouse.gov" and "stalin@kremlin.ru."

Someone asked about what led to the discovery that one of the largest lists collected contained domains for use in randomizing spam by way of templates. This led an audience member to inquire as to whether templates were linked to dictionary lists so as to better convince the receivers of the spam's legitimacy. Christian's group did not observe the behavior, but he admitted that it is an interesting possibility. Other questions related to the encrypted communication present in Storm and about the ease of infiltrating the network. The speaker noted that infiltration was surprisingly easy and encrypted communication is subject to man-in-the-middle attacks. Niels Provos wondered whether they'd tried to inject error messages to the bot master. They did not, but the question led to a discussion of how easy it would have been for the bot master(s) to detect their presence. The bot master could have asked Kreibuch's worker bots to send spam to certain addresses and then checked whether the spam was actually sent, but this did not happen.

- ***Characterizing Botnets from Email Spam Records***

Li Zhuang, University of California, Berkeley; John Dunagan, Daniel R. Simon, Helen J. Wang, Ivan Osipkov, and Geoff Hulten, Microsoft Research; J.D. Tygar, University of California, Berkeley

John Dunagan presented a work led by Li Zhuang at UC Berkeley that used trace information present in spam messages to correlate spam campaigns. Their spam corpora was gathered from the "junk" folder of Hotmail users over 9 days. Using this data, they discovered that 50% of spam botnets have more than 1,000 bots and 80% of botnets use less than half of their bots in each spam campaign. The last statistic begs the somewhat depressing question: Have spam botnets reached the point where they don't need as many bots as they have? In addition, Dunagan indicated that 60% of botnet-related spam is from long-lived botnets.

To associate spam bots with botnets, they attempted to link these bots to individual spam campaigns, in the hope that the same spam campaigns are perpetrated by individual botnets. This was accomplished by using three separate techniques. First, the same spam campaigns tend to use the same target URLs (i.e., ask the spammed user to visit the same site). The target URLs had to match exactly for this metric to work, which seems to be a somewhat defeated spam campaign correlation mechanism based on the randomization of URLs discussed in Kreibuch's presentation. Their second technique to link spam campaigns, then, used the similar body content present in messages. Finally, they also attempted to link bots to botnets based on whether the same bots are participating in the same campaigns.

Once they associated a spam campaign to an individual botnet, they tried to estimate the number of individual machines present in the botnet. This becomes difficult because of the prevalence of dynamic IP addresses among compromised machines. So, they used MSN data containing login events to link machines across dynamic IP addresses and thus to establish the variation pattern on subnets. Because users could easily be logging in from home and then from work, they define an upper bound on the potential variability present on subnets.

The first questioner asked how overlapping content in spam messages was used, given that the messages are often designed to defeat such correlation techniques. Dunagan said they used Rabin fingerprints and that currently used spam obfuscation techniques do not achieve enough polymorphism to make correlation impossible or even difficult. Another audience member asked whether the team notified MSN users found to be infected. Dunagan noted that their MSN data was not from the same 9-day period as their spam data; while they might be able to notify a user that they were infected a month ago, they didn't have the clearance to do so.

- ***Peeking into Spammer Behavior from a Unique Vantage Point***
  Abhinav Pathak and Y. Charlie Hu, Purdue University; Z. Morley Mao, University of Michigan

Abhinav Pathak presented the third and final spam talk at LEET. His research observed spam from the vantage point of open SMTP relays. To collect data, they set up an open relay that sent only those messages that test for open relays. All other email was blocked. Spammers attempting to locate open relays send messages containing the IP address of the relay they are testing to email addresses the spammers control. Thus, to fool the spammers into thinking the relay is functional, Pathak's team allows sending these messages. This methodology for convincing spammers of an open relay also leads to the relay being blacklisted by projects such as Spamhaus. To counteract this, emails containing the strings DNSBL, ORDB, and a few others are not relayed.

Their open relay data collection approach identified two types of spammers: low-volume spammers (LVS), which appear in large numbers and use coordinated spamming at a low rate and low volume, and high-volume spammers (HVS), which have fewer nodes and send uncoordinated/disorganized spam at a very high rate of throughput. The LVS are considered more interesting because of their coordinated approach. They perform open relay scanning and distribute the open relays identified. The list of email addresses is also split into chunks and processed so as to avoid sending the same message to the same address multiple times. The chunking they observed is done alphabetically and is thus easily identifiable.

Perhaps the most interesting portion of the talk came in the discussion of a graph of email list chunk number versus time. This graph allows a systematic distinction to be made between the LVS and HVS types. The LVS spam increases linearly over time whereas HVS spam happens in one burst. Also, based on the observation that list chunking happens alphabetically, the graph also allows the separation of spam into spam campaigns.

Some interesting questions centered on the effectiveness of spam blacklisting. One audience member inquired as to the effect on observed spam when Pathak did happen to get the relay blacklisted. Pathak replied that upon blacklisting their open relay, spam stopped entirely, indicating that either spam blacklists are checked by spammers or that spammers constantly test open relays for efficacy. Other audience members inquired as to the amount of spam that is actually sent using open relays, given the automatic open relay blocking by Hotmail and other large email hosts. These questions couldn't really be answered, but work is being conducted now to better grasp how much spam employs open relays.

- ***Behind Phishing: An Examination of Phisher Modi Operandi***
  D. Kevin McGrath and Minaxi Gupta, Indiana University, Bloomington

Kevin McGrath presented his measurement study on phishing. His intention was to determine whether phishing URLs have differing characteristics when it comes to URL composition, registration, and cycle. He had two data sources: Mark Monitor, which is a list of phishing sites obtained from large ISPs that are verified by hand and updated every 5 minutes, and PhishTank, which has a list of community submitted and verified phishing URLs updated once every hour. McGrath also obtained the zone files for the com, net, info, and biz top-level domains.

Their methodology for information gathering begins by obtaining a thin whois of the domain upon the domain's first occurrence. Then when the feed is updated, they fetch the DNS records for every domain seen to date to establish domain life cycle. They also perform geolocation via the IP2location service. Collecting these pieces of information over a period of 211 days allowed McGrath to establish several patterns in phishing domain characteristics. He gave details of the composition of phishing URLs. For example, over 30% of phishing domains are 8 characters in length, and the relative letter frequencies between phishing and nonphishing domains differ considerably. McGrath notes that the characters a, c, and e tend to appear with the same frequencies in phishing domains, whereas nonphishing domains follow the typical English frequency table. The more interesting observation is in the lifetime of a phishing domain, lasting approximately 3 days on average.

Someone inquired as to whether this study was really a characterization of phishing domains or whether it was simply characterizing data present in MarkMonitor and PhishTank. The answer is of course unknown as there is no global list of phishing sites, but it is an important point. An audience member also inquired about the incentive of domain name registrars to fix this problem, given that they receive money for these registrations. McGrath responded that registrars do not profit from typical phishing sites because of the 5-day registration grace period. If a domain lasts less than 5 days, no money is exchanged. This fact also yields a deeper understanding as to why the average lifespan of a phishing site is under 5 days.

NEW THREATS AND RELATED CHALLENGES

Summarized by Rik Farrow (rik@usenix.org)

Awarded Best Paper!

- ***Designing and Implementing Malicious Hardware***
  Samuel T. King, Joseph Tucek, Anthony Cozzie, Chris Grier, Weihang Jiang, and Yuanyuan Zhou, University of Illinois at Urbana Champaign

Sam King began with some history of similar attacks, as well as a mention of the recent sales of Chinese-made, bogus Cisco gear by contractors to U.S. DoD customers. King and his co-authors have designed the Illinois Malicious Processor (IMP), a SPARC v8 processor that runs Linux with a twist.

They implemented the IMP by adding a small number of gates (.05 and .08% of the total gates) in two different attacks, using a FPGA (Field Programmable Gate Array) programmed using a modified version of VHDL (Very High Speed Integrated Circuit Hardware Description Language) for the Leon3 implementation of the SPARC processor.

In one attack, a local attacker runs code that includes a sequence of bytes that gets detected by additional code in the logic of the data cache controller. When this trigger is seen, other added logic loads code and data into the L1 caches, executes this code, and elevates the privilege level of the process that sent the sequence of bytes as the trigger (instant root).

King also presented a second design, called shadow mode, where the trigger sequence appears in a dropped network packet, and the code to execute gets copied from the data portion of this packet. King described two attacks, one where login as any user is permitted with the password "letmein" after the trigger and a second that hooks read and write system calls and captures possible passwords. The login backdoor exits immediately after use, disappearing from cache, whereas the password capture code remains resident. The login attack has a small impact on performance (barely more than that of a local attacker logging in as root), but the password capture attack results in 13% loss in performance. King then demonstrated the login attack using the embedded system with the IMP version of the SPARC he had set up.

The first questions related to how easy it might be to discover this attack. Sergey Bratus mentioned that in the USSR, chips were routinely reverse engineered specifically to address this attack, and King countered by mentioning the CIA pipeline control software that was acquired by the Russians and caused a catastrophe when used. Another person wondered whether multicore processors would make this trick more difficult. King responded that the same changes could be used in all processors. Kevin McGrath suggested that special-purpose multicore systems might even make this attack simpler if you just target the one core you are interested in. Brandon Enright pointed out that the MMU or some other device might work as well, but King stated that the CPU got to see the entire dynamic instruction stream, making it better suited as a target for this attack.

- ***Catching Instant Messaging Worms with Change-Point Detection Techniques***
  Guanhua Yan, Los Alamos National Laboratory; Zhen Xiao, Peking University; Stephan Eidenbenz, Los Alamos National Laboratory

Guanhua Yan begin by explaining the issues with IM worms. Instant Messaging relies on servers for transferring messages, but the protocols permit file transfer directly between clients that a worm can use to infect another system without passing through any server. IM worms can also use a URL that points to a malware download site, also resulting in potential infection without passing through a central server.

The authors propose a statistical method that watches for the change-point in frequency of file-transfer requests or URLs being sent. They designed and tested, using simulated infections, two algorithms based on CUSUM, a sequential analysis technique used for monitoring change detection. In their simulation, their algorithms were able to detect the presence of both aggressive spreading and self-restraining IM worms. The self-restraining worms would be designed specifically to avoid detection by throttling infection attempts below a threshold.

Niels Provos asked how computationally expensive their algorithms are. Yan answered that the performance scales linearly because you can keep track of past values for total file transfers or URLs included. Provos also asked about the computational complexity, and Yan said that their algorithm is $O(n^2)$ and is practical for up to 100 internal users. Someone else observed that social intimacy in IM is very skewed, with most conversations with 1.9 buddies over a month in AIM, and 5.5 in MSN, so worm propagation could be detected more simply by noticing abrupt changes in social intimacy. Someone else asked whether all clients could become infected during the five-minute window used in the experiment, and Yan responded that only a fraction of clients were infected in five minutes. Angelos Keromytis and Niels Provos wondered whether network intrusion detection that watched for patterns in data would work as well. Yan pointed out that this approach is statistics-based. The session chair ended the discussion at this point.

- ***Exploiting Machine Learning to Subvert Your Spam Filter***
  Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I.P. Rubinstein, Udam Saini, Charles Sutton, J.D. Tygar, and Kai Xia, University of California, Berkeley

Blain Nelson proposed techniques for preventing spammers from poisoning Bayesian spam filters. Bayesian filters must be taught the difference between ham (good email) and spam. The spammers do this by creating emails that will be classified as spam, for example, by including the words "replica Rolex" in the subject, then including a large number of nonspam words into their message. The goal is to cause the spam filter to misclassify ham (nonspam), and thus force the adjustment of the spam threshold so that more spam gets through the filter. Another possible goal would be for an attacker to cause an email, for example a bid, to be misclassified as spam. For example, sending the most common 90,000 tokens from Usenet postings (a set that includes both common misspellings and slang) increases the misclassification rate to 36% when just 1% of the mail is used for training the SpamBayes to recognize spam.

The authors suggest the Reject On Negative Impact (RONI) defense, where any email message that causes the SpamBayes filter to begin to reject a set of known ham messages must not be included in the spam learning set. This approach works well against dictionary attacks, but not against focused attacks. The authors also used a second technique, in which the thresholds for ham and spam get adjusted dynamically.

Jaeyeon Jung asked about the spammer sending multiple messages instead of just one, and Nelson responded that that method results in less impact, so many more messages are required. Someone else asked whether this was why spammers were including blocks of valid text in past spam, and Nelson answered that it is not clear why spammers were doing this in the past, but if you have enough tokens, the effect would be one of poisoning SpamBayes. Another person asked about excluding just some tokens instead of entire messages, and Nelson said they hadn't looked into that, leading to some discussion. Brandon Enright suggested defending against this attack by using bigram (word pairs). Nelson answered that they were looking into doing that. Sergey Bratus wondered whether they check if the message is actually read or not in deciding on using it in training; Nelson responded that they did consider some work like this. As the workshop broke up for lunch, a small crowd gathered around Nelson.

MEASUREMENTS, UNCERTAINTIES, AND LEGAL ISSUES

Summarized by Rik Farrow (rik@usenix.org)

- ***Conducting Cybersecurity Research Legally and Ethically***
  Aaron J. Burstein, University of California, Berkeley, School of Law

Aaron Burstein began his talk with a disclaimer. "Nothing in this presentation constitutes legal advice. If this was legal advice, it would be followed by a bill." He then went on to explain the U.S. legal landscape that impacts security researchers. The DMCA (Digital Millennium Copyright Act) has no research exception, for example. For researchers interested in capturing network traffic, the relevant laws are:

- Wiretap Act: Prohibits real-time interception of the content of electronic communications; the distinction between content and noncontent is vague, with the To and From lines being noncontent, but the Subject line of email is considered content.
- Pen Register/Trap and Trace statute: Prohibits real-time interception of noncontent portions of electronic communications.
- Stored Communications Act (SCA): Prohibits providers of "electronic communications service to the public" from knowingly disclosing the contents of customers' communications.

All three of these acts include loopholes that allow the providers of a service to monitor and capture network data. In the cases of Wiretap and Pen/Trap acts, providers may capture whatever content or noncontent they want as needed to protect the "rights and property" of the operator. In the case of the SCA, the operator can use stored data within the organization however they want. But in all of these cases, handing over this data to a researcher could be illegal.

The Wiretap Act and the SCA both came before widespread computer networks, and the Electronic Communication and Privacy Act (ECPA) and Computer Fraud and Abuse Act (CFAA) were written later. Burstein then presented two scenarios. In the first, a researcher approaches a commercial ISP and asks for packet traces. Burstein points out that this would be covered by the ECPA and that there are no research exceptions. At this point, I asked about ISPs who share content and noncontent data with advertisers so the ISP can insert ads into email and Web browsing. Burstein said that this is allowed under the law. Someone else asked about having a student who works for an ISP during the summer. Burstein thought this would work, as long as the student did not remove the data from the ISP. Even continuing to use a login account to view logs later appeared to be okay.

In the second scenario, a researcher is capturing malware, allowing it to infect a sandbox, then watching what the malware does on the network. Note that this is similar to what Polychronakis et al. did in their paper, except that they prevent the malware from infecting other machines and captured all communications. Burstein said that if the researcher permits the malware to send out code and or data that infects systems not under the researcher's control, that would be in violation of the CFAA. He noted that the CFAA does not ban malware, that communicating with any external system was problematical, and sending out malware or even certain data (the CFAA specifically prohibits the sending of stolen passwords and financial data) runs afoul of the law.

Burstein concluded by saying that researchers should work closely with their own network administrators, as they can then work to help protect the rights and properties of the network owner while having legal access to network content and noncontent. He suggested both legal fixes, as well as working toward best practices and a code of conduct.

Someone asked whether a researcher has a duty to report certain content, and Burstein pointed out that the ECPA does allow you to report certain things. In some cases, such as discovering child pornography, you have an obligation to report, and running crawlers can put you into serious jeopardy.

- ***Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm***
  Thorsten Holz, University of Mannheim; Moritz Steiner, University of Mannheim and Institut Eurécom; Frederic Dahl, University of Mannheim; Ernst Biersack, Eurécom; Felix Freiling, University of Mannheim

Thorsten Holz presented more work related to Storm, and as he did so, it quickly became apparent that groups of researchers had actually been interacting via the Storm in an unexpected manner that has inflated the reported size of Storm botnets. Storm uses P2P for commands and updates, but it also communicates with a list of servers, so it is a hybrid. The P2P portion uses Overnet, and by crawling Overnet, Holz and his co-authors discovered 45,000–80,000 Storm bots at different times. They send out probes every 30 minutes, whereas the UCSD group (Kanich et al.) sends probes every 15 seconds.

Holz reported that Storm infections tripled over the Christmas to New Year week of 2007 because of successful social engineering attacks. Fabian Monrose asked why the numbers go down sometimes, and Holz replied that events such as MSRT sending out a patch can result in systems becoming clean. Then Holz stated that they introduce 2^{24} hashes (16 million) to the P2P system (the hashes being used to locate bots), and Niels Provos immediately asked whether this could inflate the number of discovered Storm bots. Holz said this certainly could, and someone else said "That's you!" Holz went on to mention that they had also experimented with disrupting Storm. One method relies on introducing sybils, malicious peers under the control of the researchers, that can be used to spy on traffic and abuse the network in other ways.

Through their crawling of P2P and their sybils, Holz claims to have seen between a minimum of 5,000–6,000 and a maximum of 80,000 Storm bots per day. David Dagon, the session chair, suggested that perhaps researchers need to set up a Storm users list. Someone else asked why they don't see the 16 million nodes represented by the hashes Holz injects into the network. Holz responded by saying they are using only two IP addresses. Someone else mentioned that researchers need to be consistent in their methods, so they aren't tripping over one another while researching Storm. Brandon Enright of UCSD (another Storm researcher) expressed concern that the Storm authors might stop using Overnet (the P2P network that Storm relies on), and Holz agreed. You can learn more about Storm from previously published articles in the December 2007 issue of *;login:*.

- ***The Heisenbot Uncertainty Problem: Challenges in Separating Bots from Chaff***
  Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, and Stefan Savage, University of California, San Diego

Chris Kanich described the UCSD team's work in determining the number of active Storm participants and succeeding in outing another researcher active in crawling/poisoning the Storm botnet. Kanich pointed out that the number of claimed Storm bots is extremely high, with MSRT reporting a lower bound of 275,000. Kanich reported that research groups, as well as competitors to the Storm botnet, have been very active, and that this has inflated the number of nodes.

Storm uses Overnet, a P2P file-sharing network based on the Kademlia DHT algorithm. The UCSD team discovered an error in the generation of unique object IDs (OIDs) used by Storm, limiting the total number of OIDs to 32k (2^{15}). This does not place an upper bound on the number of nodes, as not all nodes will communicate, but it does make the OID itself into an oracle that can identify a true Storm infection as opposed to a file-sharing client or another research crawler. The UCSD team built a tool named Stormdrain that implements a state machine for categorizing Overnet nodes. Potential Storm nodes are only considered Active when they actually respond, and nonresponding systems are moved into a Removed state, then quickly into a Dead state, over a short period of time.

Someone asked about dynamic IP address, and Kanich replied that they don't care about this, as they are only interested in the instantaneous number of nodes. Someone else pointed out that Kademlia should time out old peers, but Kanich reported that Storm's implementation is broken, and its K buckets are not recycled every four hours as they should be. David Dagon noticed a spike in a graph and asked when that occurred. Kanich replied, "March 10," to which Dagon said, "I owe you a drink." Kanich described improvements in Stormdrain, such as advertising OID hashes that are "close" to recently advertised peers, and this increased the proportion of nodes considered Active rather than just Live. Gary Warner wondered whether the Storm nodes could be distinguished from Overnet nodes based on the command set used, and Kanich replied that although they didn't do that, it should work.

During three weeks of Stormdrain crawling in March 2008, the number of active nodes varied between 8,000 and 23,000 Active nodes. David Dagon asked whether the UCSD group would be willing to coordinate with his groups in probing, and both Kanich and Brandon Enright said they would be willing to communicate with other researchers.

An article on Storm begins on page 6 of this issue.

- ***Ghost Turns Zombie: Exploring the Life Cycle of Web-based Malware***
  Michalis Polychronakis, FORTH-ICS; Panayiotis Mavrommatis and Niels Provos, Google Inc.

Michalis Polychronakis presented this paper, which expands on work published last year at HotBots about drive-by downloads. Drive-by downloads involve Web pages that have been modified to include script or iFrame sections, resulting in the installation of malware on systems, currently focused on Windows. In this work, the researchers monitored attempted communications after infection, analyzing over 448,000 responder sessions. Polychronakis said that they found that malware reports information about the infected system, address books, browser history files, stored passwords captured by keyloggers or browser hooks, and attempts to join botnets.

Their setup used Windows systems running within VMs that were passed a URL suspected of causing drive-by downloads. To capture outgoing connections, they set up a number of proxies for known protocols, as well as generic responders that often worked, even though the actual protocol was unknown. The generic responder looks for hints to the protocol when a nonstandard port is used, then emulates that protocol if known. If unknown, a generic banner gets sent to the malware if there is no activity after a number of seconds, and this often resulted in a useful response. Besides connecting to servers that collect stolen data, malware often portscanned local networks, looking for Windows services such as SMB, NetBIOS, MSSQL, and DCOM.

McGrath asked whether some requests to nonstandard ports were using HTTPS, and Polychronakis replied that they generally were not using that protocol. Jaeyeon Jung asked how many types of malware were seen; Polychronakis responded that they didn't analyze which malware family was sending traffic as they couldn't perform analysis on so many infections. Someone else asked about the capacity of their system, and Polychronakis said they could check about a million pages a day using their setup. John Ramsey mentioned they had developed Caffeine Monkey, which does some URL analysis. Then he asked whether the malware was encrypted or packed. Niels Provos answered, that most is at least obfuscated and a lot of the Javascript is encrypted. David Dagon asked whether the malware tests to see whether it is running in a VM or in an emulated environment. Provos responded that malware download servers won't even respond to requests from IP source addresses known to belong to researchers' networks. But they have not seen malware that appears to be aware that it is running within a VM.

WORK-IN-PROGRESS REPORTS

Summarized by Joshua Mason (josh@jhu.edu)

Will Drewry presented a methodology for fuzzing regular expressions. Although the methodology was not discussed in detail, their results were quite impressive. Their fuzzer has so far led to 15 security advisories, with 3 or 4 causing code execution. The impact of the methodology is intriguing because of the number of applications affected by the regular expression engines they broke. Their advisories affect applications such as Adobe's Flash Player, Apple's Safari browser, Adobe Acrobat Reader, and Postgres SQL. Adobe Flash alone is one of the most prevalent pieces of client-side software on the Internet today, with over 98% market penetration. They intend to publish the source for the fuzzer, which will hopefully lead to more secure regular expression engines in the future.

Gary Warner from the University of Alabama at Birmingham presented an ongoing work aiming to gather an unprecedented amount of spam. He presented techniques he is currently employing, such as asking for the MX record for popular domains without a mail server and voluntarily submitting their email addresses to email address farming malware. Warner's team is also attempting to get an "opt-in" plug-in for SpamAssassin that would, if a user agrees, have all the user's spam sent to their spam collection project. The intended uses for the captured spam are numerous; he briefly discussed using some data-mining algorithms to attempt day-to-day spam campaign tracking.

Rick Wesson from Support Intelligence presented an ongoing Internet mapping project. They use software from measurementfactory.com to map live portions of the Internet. The point of the project is to employ visualization tools to establish trends present online. Data gathered can potentially be used for a variety of applications, such as establishing malicious segments of the Internet.

David Dagon presented a project he's working on that he calls "memory dumpster diving." He intends to use his technique to perform automated memory analysis on malware. This would spare malware analysts from performing the arduous task of constantly having to reverse engineer new instantiations of the same general bot software to obtain required information such as encryption keys or connected hosts. His platform would perform run-time analysis to dump what seem to be relevant portions of memory, so the analyst can simply take the information he wants out of the memory trace.

Thorsten Holz presented a measurement study he and Frederic Dahl are working on that gathers data on DDoS attacks launched by the Storm worm. So far, it seems the Storm worm's attacks last an average of 90 minutes at 61 packets per second and are typically against either individual users or anti-spam/anti-spyware companies. He also very briefly covered some new reverse engineering they were able to do on the Storm networks' encrypted communication. They obtained the RSA key and can now encrypt messages to Storm nodes to make them connect to arbitrary hosts.