Interview with Stefan Savage

On the Spam Payment Trail

RIK FARROW

Stefan Savage is a professor of computer science and engineering at the University of California, San Diego. He is also director of the Center for Network Systems (CNS) and co-directs the Cooperative Center for Internet Epidemiology and Defenses (CCIED), a joint effort between UCSD and the International Computer Science Institute.
savage@cs.ucsd.edu

Rik is the Editor of *;login:*.
rik@usenix.org

I have been following, with great interest, the work of many researchers in following the spam trail. Over time, I noticed that a number of researchers were obviously working together a lot, combining their efforts into what appeared an immense task: understanding of an entire underground economy.

I was fortunate enough to find Stefan Savage, one of the primary investigators in this work, in a storytelling mood. What follows is his detailed account of successes and failures, approaches that appeared to be dead ends where students prevailed, and how we now have a large body of solid research in an area that has confounded many attempts to come to grips with its many interlocking pieces.

Stefan invited Vern Paxson and Geoff Voelker to participate in the email interview process. Both made suggestions and provided corrections to Stefan's tale, and were content not to have their contributions made explicit in this process.

If you are attending USENIX Security this year, you will find three papers related to this story, and one at CSET. And there are more than a dozen other papers that have come from this collaboration.

The Interview

[RIK] How did you get interested in assigning value to malware and spam?

[STEFAN] The truth is that Vern Paxson (UC Berkeley), Geoff Voelker (UC San Diego), and I started down this path back in 2006. We'd been working together for quite a few years on large-scale attacks (e.g., worms, viruses, DDoS, etc.), and while we'd had lots of technical successes looking at those problems head on, it was pretty clear that the world wasn't getting any more secure. Around that time we became exposed to the breadth of activity involved in underground trading of compromised accounts, credit cards, spam mailers, email lists, etc.—anything you could think of. This was really our inspiration, because we came to recognize the role that the profit motive was playing in all this (although spam was key to this evolution, we wouldn't make the link until later).

I think it helped that at the time I was reading a book on the history of the drug war and the failings of supply reduction as a strategy due to the poor understanding of drug distribution economics. We came to see that our community had a similarly poor understanding of the value chain for economically motivated attackers and thus didn't understand that our various technical interventions actually played minor roles, at best, in mitigating their actions.

During the summer of 2006, Vern had an intern up at ICSI, Jason Franklin from CMU, and we got him to focus on a big trace of underground IRC data we had gotten our hands on. The analysis was ultimately published in CCS [Computer and Communications Security conference, 2007], with Jason's advisor Adrian Perrig as a co-author [1], and while it was shallow and there was quite a bit we got wrong, I think it marked the turning point for us. From then on we started thinking much more holistically about our security work, trying in particular to understand what the underlying economic models were and how we might access those through measurement.

Around the same time (maybe just a bit earlier) we got into spam, due to a project that Geoff Voelker had started using a large spam sink. This was an old .com domain with no real users for which all the received mail was expected to be spam. David Anderson and Chris Fleizach (two master's students at UCSD) took this data and started crawling all the URLs embedded in the spam emails. They would then render any of the Web pages they found and cluster them together based on image similarity, using a technique called image shingling. The goal of that study [2] was to look at the servers being used to host the sites advertised in spam and look at the dynamics of their lifetime. Again, we didn't fully understand the subject matter, but we'd clearly found another piece of the puzzle—that one might want to consider the sites being advertised independently from the advertisements (i.e., the spam). It also helped us to build up some of the infrastructure experience that we'd need in the years to come. Finally, this infrastructure *inadvertently* got us into looking at Storm.

By the way, I should be up front that two people who definitely influenced our thinking early on were David Aucsmith (Microsoft) and Rob Thomas (Team Cymru). Dave I met through an NRC study I was on; he was the first person I'd run into who was talking about the *price* of sending spam and mounting DDoS attacks (as opposed to some technical quality like packets per second or spams per day). Rob was in this space very early, and I think several of us knew of him through different channels (via CAIDA, NANOG, etc.). We were heavily influenced by his world view, part of which got documented by *;login:* in his article [3], as well as his terminology (miscreants, underground economy, etc.).

[RIK] That article about the underground economy is still very popular today, judging by the number of downloads from the USENIX Web server. And the terminology presented was not just that invented by Rob Thomas. I had to ask him and the other authors to define many other terms, taken from the underground sites they had gained access to, like "bins," "rippers," "cashiers," and "wells," none of which matches its dictionary meaning.

Team Cymru did a lot of work by monitoring IRC and other servers. You mentioned collecting spam, and two papers where you analyzed links and Web pages that came from the collected spam. Did you do more work with your spam sources?

[STEFAN] We asked Chris Kanich, who was then a fairly new UCSD PhD student, to take over the spam feed, and we had some kind of idea about maybe trying to look at click-through rates by looking at spam-advertised URLs and then seeing if campus would let us monitor how many outbound visits went to those same domains, or something similar. In a bit of serendipity, our machine was suddenly hit by a large (at the time) DDoS attack, greater than 1 Gbps, which got campus network managers to notice. We weren't the only ones experiencing this, however.

What had happened is that the folks running the Storm botnet added a bit of logic that would profile visitors, and if a single visitor accessed too many of their sites within a particular time period they assumed that it was a security researcher and started DDoSing them—behavior activated by our spam crawler [4].

This attack caught our attention, as it did much of the research community. Indeed, I think it was this particular behavior of Storm that caused it to garner so much attention in the beginning. For a while we pursued a tangent, trying to use this behavior to measure a DDoS attack from the victim's vantage point. Consequently, with the permission of campus, we then restarted our crawler over a weekend and set up multiple packet monitors to get a full trace.

This project led to nothing, but had the serendipitous side effect of introducing us to Brandon Enright, then an undergrad working for the campus security group at UCSD. Brandon had become independently interested in Storm and had written code to crawl the Overnet Distributed Hash Table (DHT) that Storm used for coordinating its various bots. His goal was to enumerate all the IP addresses participating in the botnet at any particular time. A description of a later version of this work, and the challenges of such enumeration, later appeared in LEET [5]. Brandon was doing this to clean up Storm-infected bots at UCSD and sharing the data with others to do their own remediation, but he got our students interested in the details of how the botnet worked. Quickly a small group formed, with Brandon having the most hands-on malware experience, while Chris Kanich, Kirill Levchenko (a UCSD PhD student at the time), and Christian Kreibich (a researcher at ICSI) just started running instances of the Storm binary in a controlled environment and poking at it.

[RIK] I really liked that LEET paper. I asked Brandon and several other authors to write an article based on it for *;login:* [6].

[STEFAN] Let me give a tiny bit of background here to explain how this came to be. First, we'd had a close working relationship with Christian since 2006 when he collaborated with Kirill and Justin Ma (now a postdoc at Berkeley) on a system to automatically cluster packet traces by protocol (without a priori protocol knowledge). Christian was Vern's postdoc then, but Vern was completely open to him coming down for a couple of weeks to get this project done, and that pretty much set the stage for a silo-free group culture in which our various students and staff all feel free to work together (and tend to do so). Second, both the UCSD and the Berkeley groups had spent a bunch of time building malware containment systems—us with Potemkin [7], which was largely a research vehicle, while Weidong Cui (now at Microsoft Research) and Nicholas Weaver had built GQ [8]. Christian had then rewritten it, and that is what the group uses today. Moreover, GQ in turn benefited tremendously from Vern's investment in building Bro and related network analysis tools, so it was *reasonable* to automate the manipulation of network trace data (e.g., binpac [9], RolePlayer [10], etc.), which became important in the next year.

[RIK] I've often seen the names of a group of advisors and students from UCSD, UCB, University of Washington, and ICSI on related papers. I'm beginning to understand how this came about. It helps to see the bigger picture behind the names seen in papers, and how different researchers combine their strengths toward a common goal.

[STEFAN] Returning to the story, this little team got excited about understanding how Storm worked, but—aside from Brandon—they had basically zero skill doing

reverse engineering. So not knowing that this was a crazy approach to pursue, they tried reverse engineering the command and control (C&C) protocol in a blackbox fashion—sending data at a captive bot, writing down what it did, theorizing about why it did those things, or letting it talk to its normal C&C and seeing what it tried to do in response to various commands it received. Brandon was busy, but provided key insights when they hit roadblocks (e.g., message encryption), but the rest was just raw guesswork over a period of several months. Vern and I had our doubts whether this was a good way for everyone to spend their time, since we weren't confident they could do it, or even what the research question would be if they succeeded. Geoff Voelker was on sabbatical in India for this period, so he was blissfully unaware of how much time was being wasted on this. However, we gave the students a long leash and somehow they pulled it off, documenting most of the C&C protocol and then building a set of parsers that could interpret it.

Once we realized how much information was contained—for example, how the spam messages were encoded within polymorphic templates, who the spam was being sent to, the delivery success rate, etc.—we realized we had a unique opportunity to look at how spam distribution worked from the standpoint of a botnet operator. We did a quick passive characterization of this data, which became the "On the Spam Campaign Trail" paper from LEET '08 [11]. As soon as we started writing it, however, Kirill pointed out that knowing how to parse Storm's C&C was also equivalent to being able to inject or change C&C commands. This would lead to our first real economics study. To give a bit more context, one needs to understand a bit about how Storm was structured circa late 2007–early 2008.

The basic Storm infrastructure was divided into three tiers: "worker bots," which were responsible for sending spam email or mounting DDoS attacks; "proxy bots," which provided public points of connection for worker bots; and master servers who provided commands to (and received feedback from) workers via the proxy tier. Workers and proxies were built out of compromised hosts and automatically differentiated based on whether they had external IP connectivity, allowing them to act as proxies versus workers. The master servers were dedicated machines in datacenters, such as Intercage, a California-based Web site hosting provider. Workers would select a quasi-random proxy using the Overnet DHT protocol and would then send, effectively, requests for work, which the proxy would then forward on to the master servers and similarly forward the responses of the master servers back to the workers. Proxies had some master server locations hardcoded and could received signed updates indicating the location of other such servers.

[RIK] I learned much of this by reading the LEET paper [5] and Brandon's *;login:* article [6].

[STEFAN] So, using a sample of the Storm malware, it was relatively easy to infect a machine and have it "become" a proxy and communicate with workers and master servers—just as a real infected host would (using our previous honeypot experience to carefully wall it off from accidentally sending email or DDoS attacks). Moreover, by building code to parse the messages as they went by, it was possible to actually change the responses being provided by the master servers in real time... in effect leaving the underlying process in place but manipulating one component. In particular, we could modify the URLs that the master servers provided to worker bots to be included in their outbound spam messages and have these point to sites under our control.

[RIK] I guess that this is the point where you needed to talk to lawyers?

[STEFAN] Yes, this is where we first started talking to our lawyer friends in depth. While the students were off making the capability a reality, we engaged with the people we knew who were best versed in Internet legal issues (e.g., the Computer Fraud and Abuse Act, the Electronic Communications Privacy Act, and the CAN-SPAM legislation) to help us figure out if we could actually do this. The first thing you find out when you start asking legal questions in this space is that no one can tell you "X is legal," nor is there any government agency who is authorized to certify such an effort. Even something as simple as sending ping packets to random hosts does not have "cut and dried" legality. Lawyers and legal scholars can frequently tell you whether something is clearly illegal, but if not, it's all about understanding the risk profile and working up the legal theory under which one operates. This took us quite a bit of time and we pursued multiple opinions to make sure there was agreement. In the end, while this area is rife with risk, the very specific circumstances around how Storm operated (e.g., being pull vs. push, using an existing DHT network, the kinds of information being sent, etc.) created a stage on which our advisors felt it was safe to proceed. Moreover, we developed a basic set of ethical principles to determine what could and couldn't be done in the study (based on consequentialism, the idea was that our intervention should defensibly cause no additional harm when compared to an alternate universe in which we had done nothing). This did indeed keep us from doing things that we had considered. For example, we had broken the private key for Storm's master server advertisements and we had the capability, in principle, to take over the entirety of the botnet.

Having addressed these issues, we dove into creating our experiment. We came to recognize that the most interesting questions revolved around the underlying economic model for spam: how many messages must a spammer send to get a sale; i.e., how often do people actually purchase? This determines the profitability of each spam message and implicitly drives the amount of spam being sent. Conversely, it also sets a lower bound that spam filters must reduce in order to make spam unprofitable.

It was Kirill Levchenko who first devised the pipeline metaphor that we would use in the paper [12], in which large numbers of messages are sent and then discarded at multiple filter tiers (e.g., rejected by mail servers, by spam filters, by mail readers, by site visitors who decide not to buy, etc.) until the final true purchases that monetize the entire activity get through. The basic experiment was simple: we'd change the URLs on the spam email templates that traversed our proxies and have them specify Web sites we controlled. We could then compare the number of spam messages each worker attempted to send with the number of visitors we received at the site. Further, if we duplicated the sites being advertised, we could further capture how often users tried to put particular items in their shopping carts and checked out. Since Storm was sending pharmaceutical spam (advertising for affiliates of the Glavmed "Canadian Pharmacy" program) we replicated their site in great detail. Then we started.

This is where we first started to get into trouble. First, we needed to acquire domain names to be used in this study. We simply bought a bunch from GoDaddy and started using them. This resulted in large numbers of complaints being directed to GoDaddy (since some subset of people receiving spam are technically sophisticated enough to identify the registrar of the site being advertised and motivated enough to send in their complaints), who in turn started suspending our domains and sending us various challenges/threats. We regrouped and found a different registrar whom we knew personally (really a reseller of Tucows),

but this just added an additional layer in the chain of complainants. We briefly considered buying domains from ESTDomains (who at the time was a well known registrar used by criminal actors and who appeared to exert little oversight), but we decided this was a bridge too far for us. Instead, I had a surreal phone call with the fraud abuse group at Tucows to try to get their support. In trying to explain that domains we had registered were to appear in spam, but we were not sending the spam and that this was part of a research study, the first comment I received was, "You've got to be kidding me. This is the best story you could come up with?" However, after almost two hours of explanation, pointing them to past papers and mutual acquaintances to establish bona fides, the group over there realized that we weren't making it up. In the end, they thought it was pretty cool and agreed to allow us to proceed.

Our next problem was even more inadvertent. Storm also tried to infect hosts via social engineering ("Your friend sent you a card, click here to get it," sending you to a supposed eCard Web site that would provide an EXE containing the Storm binary). We also decided to replicate this using another replica site, but the binary we offered effectively did nothing (it simply reported that it had run, and even this behavior was automatically disabled if the date was later than our study period). Interestingly, AV signatures for our EXE soon appeared from most vendors (a clear indicator that the malware load had increased to a point that it was not possible to do any meaningful analysis on sample binaries). This was expected and, indeed, was ideal for our study since we wanted to—as much as possible—simulate the experience of Storm's operators (i.e., if our binary ran, it was in spite of AV and OS warnings not to do so, or indicated that users had no such security resources). However, this was the first time we had done something like this and we had not fully internalized that, in performing this infiltration, we were ourselves being monitored by others. And this is where things started to get squirrelly.

We had previously had contact with the FBI special agents in charge of investigating Storm and we had given them what insights we had. However, we had not thought to tell them that we were advancing our experiment to the next stage (i.e., changing links and setting up our replica sites). The consequence of this is that other investigators found our binary (originating out of UCSD) and concluded that we were potentially involved in working with the Storm operators. This in turn embarrassed our contact who had vouched for us, and now we looked like double agents. In the end, it was all resolved (indeed, at a meeting at the first LEET), and we learned an important lesson about communication, but we were told that, in the meantime, legal documents had been drawn up in anticipation of raiding the department's machine room and seizing our cluster.

There were other hiccups here and there, but by and large, the paper was a dream to write. In spite of its tremendous complexity, we made very few mistakes in the methodology. The only clear remaining issue was that we did not appreciate how quickly real spammers throw away spam-advertised domains (that then redirect to other sites) to mitigate the impact of blacklists. While we indeed used multiple different domains over time, ours were much longer lived, and thus blacklisting undoubtedly caused us to underestimate the response and conversion rate that the real spammers probably experienced. However, the broad results were quite clear: 75% of bot-originated spam was being immediately dropped on the floor, most of the remainder was filtered by spam filters, and only a very small fraction of users actually clicked on the links contained in such messages and an even smaller fraction ever decided to place an order. Yet in spite of this it was clear that the raw

volume of this activity could produce significant revenue. This, in turn, would lead us to wonder about the composition of the spam value chain, who made the profit, which parts were weak, etc., but this was still some time away.

The next immediate concern was one of perception. Even though our paper was in submission (to CCS) and not public, many people seemed to have copies of it (people not on the PC) and more people still seemed to "know all about it." Indeed, one close colleague called me up from a conference and said, "I wanted to let you know that everyone is talking about your paper and a bunch of it isn't positive. Someone was talking to a group here and he says you guys are going to get the whole community in trouble."

Now, normally this isn't something we care much about. However, it was exacerbated by a contemporaneous factor. During this same period there was a big public to-do caused by Chris Soghoian's CNET blog entry [13] opining that the Colorado/Washington Tor exit node study in PETS constituted a breach of civil and criminal law and moreover represented a fundamental ethical violation because there had been no human subjects review. Now, while the Tor study issue was completely overblown (it was quickly resolved and no one was sued, arrested, or even censured), the underlying concern about oversight was real; it was clearly a wake up call to the security community about the human subjects issue. Indeed, little of the networking, systems, or security communities knew much about IRBs or even thought in those terms at the time. We were no exception. So Vern and I spent a bunch of time reading up, getting advice, and then writing a post-hoc human subjects proposal for our study with an explicit mea culpa to the IRB that we'd already done most of it and could we keep doing this study and keep the data. This took a very long time to get through the process (one of the challenges of a multi-university study), but ultimately all of our work and use of the data was approved without additional conditions. We also made a point to include an explicit section in our published paper on the underlying ethical issues and our justification for them—a practice that we continue to this day when the issues are non-obvious.

[RIK] Hmmm, this explains a lot about why I often hear you ask other researchers whether they bothered to get IRB approval. So what happened next?

[STEFAN] Ironically, in spite of our trepidation, we received little pushback from the community when the paper was published, and the work appears to have been widely appreciated. Indeed, part of what happened is that circumstances driven by other researchers eclipsed us, and while our work had once been "on the edge," it was now being highlighted by Marc Dacier in his CACM foreword for its "great care addressing the legal and ethical issues linked to the measurement."

For us, the immediate impact of the spamalytics study [12] is that it became *much* easier to get data from partners. In some sense, it was the reputation this work built with industry that planted the seeds that would support the next two years of activity.

[RIK] It seems like your Click Trajectories paper [14] at Security and Privacy in Oakland (2011) represents another chapter in this story.

[STEFAN] The "click trajectory" effort started a bit over two years ago (although the project name came much later). At that time we were starting to get quite a bit more spam data (10 distinct feeds at the peak from various anti-spam companies and honeypots), and Kirill Levchenko was tempted by the siren song of large-scale data mining. His view was that we should be able to cross-correlate all the data and

create one of those TV movie FBI pictures with all the various participants linked by dependency arrows (in our case, botnets, spammers, fast flux clouds, registrars, affiliate programs, etc.)—the total picture of the spam ecosystem; who is responsible and where the weak points are in the business model. I think we were flush from the success on the spamalytics effort and really had no idea how much we were about to bite off.

The first big issue was how to collect additional data from our various spam feeds (sometimes millions of messages per day), including all the DNS data, registrar data, hosting data, Web page contents. Trying to bring back the old spamscatter infrastructure was a bust. It simply couldn't handle the load that we wanted to put on it and it was never designed for production use (nor did it record lots of the things we cared about). We also needed a place to put all these data that we could then make sense of. We decided to do everything from scratch.

At the core was the database. Kirill in particular had convinced us all that databases were good (all of the spamalytics work had been done using a database) which had a number of very cool side effects. First, it made certain questions very quick to answer (e.g., how many messages were sent to addresses of a particular form) and, as important, it made analyses easily repeatable. It has now become common for us to check in SQL query statements in our papers (as comments) along with the results. That way if we want to change something, we know exactly what the original query was and we can modify it without worrying if we're following the same methodology.

However, the data in spamalytics was modest by comparison. Moreover, for the click trajectory effort *everything* went through the database, because it was not only the store for final results, but also it was the trigger for additional measurements. We'd post-process raw spam emails and insert the links into "feed tables" which would be processed and then used to drive the various crawlers that would, in turn, put their results back into the database. We went through many versions of the database, killing mySQL and quickly going to Postgres, buying increasingly beefy hardware (the current core trajectory DB runs on 12 cores with 96 GB of memory, has multiple replicas, and manages a range of BLOBs in other servers, together comprising almost 100 TB of raw storage in total), and redesigning the database schema *many* times. Poor Kirill was constantly promising us that "things will be better in the next version of the schema." In the end, we needed to become very good at DB administration and optimization. UCSD PhD student Andreas Pitsillidis became that expert, through blood and sweat. In fact, about nine months ago, everyone else gave up trying to understand the full complexity of the DB system: only Andreas really gets it. While everyone did their part on this project (we had 15 authors on the final paper, all of whom made significant contributions), it was Andreas who ultimately made this all come together—I can't overstate the extent to which we could not have done this without him. Moreover, without the database (or equivalent technology) it would have been impossible to manage and process all the data we were collecting.

While the database was at the core, there were many moving pieces that fed it. First, the raw data feeds needed to be managed and normalized (and each of our data providers had their own favorite way of providing the data). Chris Kanich at UCSD became "feedmaster" (in addition to his other critical tasks) and dealt with the partners, created visualizations of the various feeds, and managed the ongoing relationships with feed providers.

The other source of feeds was from the GQ honeyfarm mentioned earlier. The honeyfarm ran network-neutered instances of major spam bots so that we could observe what spam they were being commanded to send. Keeping these bots going (and the honeyfarm itself) was a major endeavor. Christian Kreibich, with help from Chris Kanich, did most of this in the beginning, but eventually Chris Grier (then a new postdoc at Berkeley) took over the operational component (to everyone's relief), since it was also core to the next big project, his investigation of the pay-per-install market.

The other challenge here was to get the latest samples of new spam bots. Here we got help from lots of people, but in the end the go-to person was Brandon Enright (a long-time collaborator working for the campus networking and security organization at UCSD) who marshaled both his own private honeypot infrastructure and his considerable connections in the community to get whatever samples we needed. This gave us some "ground truth" about which botnets were advertising which URLs (allowing us to account for issues such as the Rustock botnet's spamming of random .com URLs to poison or overload blacklists).

After the raw feeds we had the crawlers. We had several implementations of a DNS crawler that would investigate each domain name we received and find its NS and A records. Over time, we learned that we needed to explore this space more completely to extract all the alternate answers being given due to fast-flux and CDNs (creating a name hosting "cloud" for each domain). Moreover, the load became large over time, and ultimately the crawler was rewritten from scratch by He "Lonnie" Liu (a first-year PhD student) to keep up. This particular artifact was remarkable because it is the only piece of infrastructure that we've built that "just worked." It never crashed, it never gave garbage data, it seemed to scale forever, and it was never the source of complaints from other members who depended on it. Lonnie never needed to say much at our weekly status meetings.

The Web crawler was a different story. I remember Geoff Voelker and I figured, "Hey, it's just crawling. How hard can it be?" We completely misunderstood the technical challenges in scaling up to large numbers of browsers and simulating associated humans. The poor recipient of our imperfect wisdom was Neha Chachra, also a first-year PhD student, who got handed the task of making a scalable Web crawler. She started by using an open source project called Selenium (designed to automate multiple Firefox instances) for the first version of the crawler, but we had no end of problems trying to get the features we wanted to work (grabbing raw page DOMs, screenshotting, inserting clicks, etc.) while synchronizing across large number of instances.

Ultimately, Neha wrote her own controller (with energetic help from Chris Grier for low-level Firefox-fu) that spawned and synchronized thousands of Firefox instances across a cluster of machines. Over time there were many changes to the crawler to handle various kinds of automated redirects, crash recovery, simulated user clicks, and so on (usually to deal with some crazy challenge that spammers had introduced). Even more significant, we discovered that many of the large hosting platforms used by spammers would blacklist our IP addresses if we visited too many times (ultimately blacklisting an entire /24). We acquired a broad range of diverse address space (Chris Grier put this together), and the Web crawler would schedule requests through proxies to these different blocks so we could see what a normal user would see. Neha went from implementing what we thought was a minor component of the system to becoming a central point of dependency for

virtually everything (I'm sure she forgives us by now). Having so many parts, the crawler was constantly in revision; it was only recently that it became truly stable.

So, what to do with all these Web pages? Well, cluster them of course. The idea is to cluster all the URLs that lead to Web pages that are basically the same. However, the difference between "basically the same" and "exactly the same" hides quite a small nightmare. We tried quite a few different techniques. Early versions used a technique based on HTML structural features that Justin Ma came up with, and we experimented with SIFT and GIST-based visual features, but in the end we used a simple q-gram metric (how many sequences of length q are identical over some window) that worked incredibly well except for pages that were entirely based on images.

Clusters were useful for visualizing the data, and Andreas Pitsillidis created a great reporting interface that let us look at the relationship between particular groups of similar Web pages, their name server hosting, Web server hosting, the feeds we received them from, and so on. However, the real reason for clustering is that we operated under the assumption that if two pages look the same then they are probably part of the same "affiliate program," and this was key to our subsequent analysis.

Here it's worth taking a small digression to explain that modern spam is basically outsourced advertising. The spammers do not themselves sell any products but work on a commission basis for an affiliate program that handles payment processing, fulfillment, and customer service. Hosting of content and name services can be handled by the spammer or by the affiliate program, depending on circumstances (e.g., advertising based on search engine optimization, or SEO, is typically hosted by the program). Moreover, the actual spam delivery may itself be subcontracted from the spammer to a botnet operator, depending on the situation. However, these facts were not just assumptions. We spent quite a bit of time trolling around on underground forums trying to understand what we were dealing with. I did much of this work in the wee hours of the morning (as the group will attest from my random 2 a.m. ramblings about each new "discovery"), and Kirill would help when Google Translate barfed too badly on the Russian translation (many of the big programs are run by Russian speakers). Along the way we managed to acquire the "source code" for the e-shops from two of the largest pharma programs, Glavmed and RX-Promotion, which gave us ground truth about how different "storefronts" might all map to the same affiliate program. Moreover, via the broad underground marketplace, we were able to identify most of the other major programs. When we ran into a wall, Damon McCoy, a CIFellow postdoc, was the go-to person to hunt down a program.

This led to the development of another major element of the project: the tagger. The tagger is basically an oracle that looks at the HTML for a Web page and determines (1) what it is selling and (2) for which affiliate program. The first problem is easy, particularly because we don't care about false positives. We had decided to focus on pharma/herbal, luxury replicas (e.g., Rolex) and software—as these were the most spammed product categories (actually gambling and porn probably beat out software, but we had decided not to do either of those for institutional reasons), and we just checked to see if the Web page included any associated brand names (e.g., Viagra and Cialis for pharma, Rolex and Movado for replicas, and so on). This worked quite well; for example, the number of pharmacy pages we didn't classify as being in the pharma class was vanishingly small (typically these would be "image-

only" redirect pages). However, classifying which program was advertising the page was quite a bit harder.

Tristan Halvorson, yet another first-year PhD student, got pressed into service generating regular expressions based on example pages I would find for each program. I'd gotten to the point where I could recognize most programs on sight, but Tristan had to somehow render this into code. So he'd try to capture what I was recognizing, then tag the whole corpus with affiliate names. I'd go look through it and find errors, and then we'd repeat. It's really hard to describe how much work this was. I looked at easily several tens of thousands of pages over the course of the project. I still remember Vern asking me late one night, "So how did we validate the tagger?" to which I replied, "Manually." He said, "Yeah, but really, that wouldn't scale." He was right in principle, but in practice Tristan and I (with Geoff lending a hand) just spent days at it—scaling be damned. This is not an approach we'll repeat again, however. We've had another student build a supervised machine learning tool to do this that seems to do almost as well with much less effort, so hopefully that's the future.

The last big component was purchasing. We really wanted to do the end-to-end analysis—where the spam came from to where it was fully monetized, and this meant purchasing goods and receiving them. This created a whole host of problems. First, we needed the university to permit it. You can imagine the conversations: "We need to make credit card purchases from criminals for goods that we may not get. Oh, and it's entirely possible that there will be fraud directed against these cards." I still remember questions like, "Why can't you just use a purchase order?" This took at least a year of education, negotiation, explanation, documentation, pleading, and much passing of the buck before we worked it all out. The purchase phase involved huge amounts of oversight, including by our own lawyers, university general counsel, and the systemwide office for research compliance. Finally, however, a few key people at UCSD (and, perhaps more importantly, at the UC Office of the President) came through for us and gave us the approval we needed.

The next problem was where to get these credit cards. Prepaid gift cards seemed like the ideal instrument. They get processed exactly like Visa and Master Card, and you can purchase them on demand, in bulk. Plus, you can set the name and home address as you like. It was too good to be true, unfortunately. First, most of these cards had no way to get a statement: Was a charge placed on the card, for how much, and who did they claim to be? Instead, they had phone support, where you could call in to get information. We did find a small number of such cards that had an online Web statement interface and so we placed an order for a few thousand dollars' worth of these. However, we discovered that the statement didn't include the Acquirers Reference Number (this is the 23-digit number you may find on your personal credit card statement) which identifies the Bank Identification Number (BIN) for the bank acting on behalf of the merchant in the credit card transaction. Without this we wouldn't know what bank was being used and we'd need to trust the information in the merchant identification string (which is routinely false, in our experience). We tried calling in to get this information, but it was very slow going, in part because the call center was staffed by only a few people and they grew suspicious at the large numbers of calls they kept getting from us.

Using our personal networks in the security community, we did manage to find investigators we knew who had done some similar work, and they identified for

us the one card that had all the properties we desired: a Web interface and online access to the ARN number for each transaction. Ironically, it was the store brand at the Ralph's supermarket near us. It was perfect.

Then the Credit Card Reform Act passed. As part of this, the Department of the Treasury instituted a rule requiring suspicious transaction reporting on foreign transactions for prepaid cards (precisely because such cards are perfect for money laundering). The added reporting overhead made most providers just stop offering international transactions (go read the fine print on the pre-paid gift cards at your local supermarket), which included the bank sponsoring Ralph's cards. Sigh. At this point I gave up and decided we'd simply have to do without.

Thankfully, Chris Kanich hadn't given up hope. On his own initiative, he started cold-calling credit card issuers explaining the service we needed. Amazingly, he found a company who was game to help us and then negotiated a contract. One day Chris came in and said, "I think I got the credit cards." It turned out to be a spectacular resource: for a modest fee, new credit cards were created on demand including detailed information (the BIN, the card acceptor ID, the country code, and so on, far more than we ever hoped to get) on each authorization or settlement transaction of interest. Over the course of our studies, Chris and Damon ran our purchasing operation, using hundreds of different cards, email accounts, and a bevy of Google voice phone numbers that redirected to a few "burner" phones they each carried.

Surprisingly, getting these orders to properly clear was non-trivial and we had to reverse engineer components of their fraud detection system (e.g., using co-located IP addresses to source purchases, non-free emails, etc.), plus Chris and Damon needed to handle a constant stream of follow-up confirmation calls from the affiliate program's customer service arms. On top of that, managing all the raw credit card transaction data and keeping it in sync with the associated Web site data was a major time sink. Here we made the mistake of trying to make due with a large Google Docs spreadsheet, a decision we're still paying for.

These were the major pieces, but there were countless details I skipped in this description: for example, Mark Felegyhazi's whois crawler and the cross-DNS matching work that Nick Weaver did in the 11th hour.

I also skipped an adequate description of all of our failures. First, we failed repeatedly to wrap our minds around this paper. We had at least two aborted attempts to submit a paper only to discover that we still didn't really understand what we were doing. I know that Vern, Geoff, and I all had doubts if this thing would ever come together (18 months of work without anything to show can shake even the most confident person). We tried, but ended up failing, to incorporate a strong analysis of the spam delivery component (which programs were advertised by which botnets, which used Webmail, etc.), and we spent months building complex models for inferring the different individual affiliates of different program,s ultimately to discard them for the final paper. There is at least another paper's worth of work in all the stuff that we left on the "cutting room floor," but we chose to focus on the parts we were the most confident about.

For the paper submission there were a few major turning points. One was a meeting where we came up with the conceptual model of the spam value chain as comprising advertising (spam delivery), click support (translating a recipient's click into a Web site), and realization (payment processing and fulfillment). This

model, beautifully illustrated by Christian in the paper [14], gave us a way to focus on the problem. It also led to us choosing to focus our analysis on the challenges of intervening at any given place in the value chain (this had always been a goal, but originally just one among many). The other major event is when the credit card data first started coming in and we realized that there were really only a handful of banks involved in processing money for spam-advertised programs. We'd hypothesized that this might be true, but with the data in hand we knew we had a great story. Finally, in the week before submitting the paper, most of the ICSI folks came down to UCSD and everyone pushed hard to get everything done. That was an amazing time and huge amounts of work got done with everyone pitching in. This is also one of those papers where the final paper actually differs in non-trivial ways from the submission. We used the time we had to really tie up loose ends and polish the analysis. I think we all knew that this was going to be one of our important papers and everyone put in the time to make it crisp.

It also kicked up a half-dozen other projects that we're working on as we speak, including several papers to appear at CSET '11 [15] and USENIX Security '11 [16].

The one 10,000-foot thing that I really hope comes out is that our core approach is to try to understand these issues from the standpoint of the attacker rather than simply from the standpoint of the victim. I think we frequently hamstring ourselves in the security community with the notion that the adversary is some abstract and arbitrary entity, whereas frequently the adversary is concrete and has very specific goals. Understanding these goals (particularly those focused on profit-making) then lets us consider defense as a form of offense: What security investments can I make that will maximally undermine the adversary's goals? Absent this kind of analysis we end up just blindly treating random symptoms of the problem, rather than focusing on the core drivers.

References

[1] Jason Franklin, Adrian Perrig, Vern Paxson, and Stefan Savage, "An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants," *Proceedings of the ACM Conference on Computer and Communications Security*, Alexandria, VA, October 2007: http://www.icsi.berkeley.edu/pubs/networking/miscreant-wealth.ccs07.pdf.

[2] David S. Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M. Voelker, "Spamscatter: Characterizing Internet Scam Hosting Infrastructure," *Proceedings of the 16th USENIX Security Symposium*, August 2007: http://www.usenix.org/events/sec07/tech/full_papers/anderson/anderson.pdf.

[3] Rob Thomas and Jerry Martin, "The Underground Economy: Priceless," *;login:*, vol. 31, no. 6, December 2006: http://www.usenix.org/publications/login/2006-12/openpdfs/cymru.pdf.

[4] Don Jackson, "Analysis of Storm Worm DDoS Traffic," Sept. 11, 2007: http://www.secureworks.com/research/blog/index.php/2007/09/12/analysis-of-storm-worm-ddos-traffic/.

[5] Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, and Stefan Savage, "The Heisenbot Uncertainty Problem: Challenges in Separating Bots from Chaff," First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '08), April 2008: http://www.usenix.org/event/leet08/tech/full_papers/kanich/kanich.pdf.

[6] Brandon Enright, Geoff Voelker, Stefan Savage, Chris Kanich, and Kirill Levchenko, "Storm: When Researchers Collide," *;login:*, vol. 33, no. 4, August 2008: http://www.usenix.org/publications/login/2008-08/openpdfs/enright.pdf.

[7] Michael Vrable, Justin Ma, Jay Chen, David Moore, Erik Vandekieft, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage, "Scalability, Fidelity, and Containment in the Potemkin Virtual Honeyfarm," *Proceedings of the 20th ACM Symposium on Operating System Principles (SOSP)*, Brighton, UK, October 2005: http://cseweb.ucsd.edu/~savage/papers/Sosp05.pdf.

[8] Weidong Cui, Vern Paxson, and Nicholas Weaver, "GQ: Realizing a System to Catch Worms in a Quarter Million Places," ICSI technical report TR-06-004, September 2006: http://www.icir.org/vern/papers/gq-techreport.pdf.

[9] R. Pang, V. Paxson, R. Somer, and L. Peterson, "binpac: A yacc for Writing Application Protocol Parsers," *Proceedings of the 2006 Internet Measurement Conference*, October 2006: http://conferences.sigcomm.org/imc/2006/papers/p29-pang.pdf.

[10] W. Cui, V. Paxson, N.C. Weaver, and R.H. Katz, "Protocol-Independent Adaptive Replay of Application Dialog," *Proceedings of the 13th Symposium on Network and Distributed System Security (NDSS 2006)*, February 2006: http://research.microsoft.com/en-us/um/people/wdcui/papers/roleplayer-ndss06.pdf.

[11] Christian Kreibich, Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson, and Stefan Savage, "On the Spam Campaign Trail," First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '08): http://www.usenix.org/events/leet08/tech/full_papers/kreibich/kreibich_html/.

[12] Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Vern Paxson, Geoffrey M. Voelker, and Stefan Savage, "Spamalytics: An Empirical Analysis of Spam Marketing Conversion," *Proceedings of the ACM Conference on Computer and Communications Security,* Alexandria, VA, October 2008: http://www.cs.ucsd.edu/~savage/papers/CCS08Conversion.pdf.

[13] Chris Soghoian, CNET, "Researchers Could Face Legal Action for Network Sniffing," July 24, 2008: http://news.cnet.com/8301-13739_3-9997273-46.html.

[14] Kirill Levchenko, Andreas Pitsillidis, Neha Chachra, Brandon Enright, Mark Felegyhazi, Chris Grier, Tristan Halvorson, Chris Kanich, Christian Kreibich, He Liu, Damon McCoy, Nicholas Weaver, Vern Paxson, Geoffrey M. Voelker, and Stefan Savage, "Click Trajectories: End-to-End Analysis of the Spam Value Chain," *Proceedings of the IEEE Symposium on Security and Privacy*, May 2011: http://cseweb.ucsd.edu/~savage/papers/Oakland11.pdf.

[15] Chris Kanich, Neha Chachra, Damon McCoy, Chris Grier, David Wang, Marti Motoyama, Kirill Levchenko, Stefan Savage, and Geoffrey M. Voelker, "No Plan Survives Contact: Experience with Cybercrime Measurement," Fourth Workshop on Cyber Security Experimentation and Test (CSET '11), USENIX, August 8, 2011.

[16] Chris Kanich, Nicholas Weaver, Damon McCoy, Tristan Halvorson, Christian Kreibich, Kirill Levchenko, Vern Paxson, Geoffrey M. Voelker, and Stefan Savage, "Show Me the Money: Characterizing Spam-Advertised Revenue," *Proceedings of the 20th USENIX Security Symposium*, August 8–12, 2011: http://www.usenix.org/events/sec11/tech/full_papers/security11_proceedings.pdf.