The Enemy Within

WHEN THE CONFICKER COMPUTER “WORM” WAS UNLEASHED ON THE WORLD IN NOVEMBER 2008, CYBERSECURITY EXPERTS DIDN’T KNOW WHAT TO MAKE OF IT. IT INFILTRATED MILLIONS OF COMPUTERS AROUND THE GLOBE. IT CONSTANTLY CHECKS IN WITH ITS UNKNOWN CREATORS. IT USES AN ENCRYPTION CODE SO SOPHISTICATED THAT ONLY A VERY FEW PEOPLE COULD HAVE DEPLOYED IT. FOR THE FIRST TIME EVER, THE CYBER-SECURITY ELITES OF THE WORLD HAVE JOINED FORCES IN A HIGH-TECH GAME OF COPS AND ROBBERS, TRYING TO FIND CONFICKER’S CREATORS AND DEFEAT THEM. THE COPS ARE FAILING. AND NOW THE WORM LIES THERE, WAITING …

By Mark Bowden

IMAGE CREDIT: ALEX OSTROY

THE FIRST SURPRISING thing about the worm that landed in Philip Porras’s digital petri dish 18 months ago was how fast it grew.

He first spotted it on Thursday, November 20, 2008. Computer-security experts around the world who didn’t take notice of it that first day soon did. Porras is part of a loose community of high-level geeks who guard computer systems and monitor the health of the Internet by maintaining “honeypots,” unprotected computers irresistible to “malware,” or malicious software. A honeypot is either a real computer or a virtual one within a larger computer designed to snare malware. There are also “honeynets,” which are networks of honeypots. A worm is a cunningly efficient little packet of data in computer code, designed to slip inside a computer and set up shop without attracting attention, and to do what this one was so good at: replicate itself.

Most of what honeypots snare is routine, the viral annoyances that have bedeviled computer-users everywhere for the past 15 years or so, illustrating the principle that any new tool, no matter how useful to humankind, will eventually be used for harm. Viruses are responsible for such things as the
spamming of your inbox with penis-enlargement come-ons or million-dollar investment opportunities in Nigeria. Some malware is designed to damage or destroy your computer, so once you get the infection, you quickly know it. More-sophisticated computer viruses, like the most successful biological viruses, and like this new worm, are designed for stealth. Only the most technically capable and vigilant computer-operators would ever notice that one had checked in.

Porras, who operates a large honeynet for SRI International in Menlo Park, California, noted the initial infection, and then an immediate reinfection. Then another and another and another. The worm, once nestled inside a computer, began automatically scanning for new computers to invade, so it spread exponentially. It exploited a flaw in Microsoft Windows, particularly Windows 2000, Windows XP, and Windows Server 2003—some of the most common operating systems in the world—so it readily found new hosts. As the volume increased, the rate of repeat infections in Porras’s honeynet accelerated. Within hours, duplicates of the worm were crowding in so rapidly that they began to push all the other malware, the ordinary daily fare, out of the way. If the typical inflow is like a stream from a faucet, this new strain seemed shot out of a fire hose. It came from computer addresses all over the world. Soon Porras began to hear from others in his field who were seeing the same thing. Given the instant and omnidirectional nature of the Internet, no one could tell where the worm had originated. Overnight, it was everywhere. And on closer inspection, it became clear that voracity was just the first of its remarkable traits.

Various labs assigned names to the worm. It was dubbed “Downadup” and “Kido,” but the name that stuck was “Conficker,” which it was given after it tried to contact a fake security Web site, trafficconverter.biz. Microsoft security programmers shuffled the letters and came up with Conficker, which stuck partly because ficker is German slang for “motherfucker,” and the worm was certainly that. At the same time that Conficker was spewing into honeypots, it was quietly slipping into personal computers worldwide—an estimated 500,000 in the first month.

Why? What was its purpose? What was it telling all those computers to do?

Imagine your computer to be a big spaceship, like the starship Enterprise on Star Trek. The ship is so complex and sophisticated that even an experienced commander like Captain James T. Kirk has only a general sense of how every facet of it works. From his wide swivel chair on the bridge, he can order it to fly, maneuver, and fight, but he cannot fully comprehend all its inner workings. The ship contains many complex, interrelated systems, each with its own function and history—systems for, say, guidance, maneuvers, power, air and water, communications, temperature control, weapons, defensive measures, etc. Each system has its own operator, performing routine maintenance, exchanging information, making fine adjustments, keeping it running or ready. When idling or cruising, the ship essentially runs itself without a word from Captain Kirk. It obeys when he issues a command, and then returns to its latent mode, busily doing its own thing until the next time it is needed.

Now imagine a clever invader, an enemy infiltrator, who does understand the inner workings of the ship. He knows it well enough to find a portal with a broken lock overlooked by the ship’s otherwise vigilant defenses—like, say, a flaw in Microsoft’s operating platform. So no one notices when he slips in. He trips no alarm, and then, to prevent another clever invader from exploiting the same weakness, he repairs the broken lock and seals the portal shut behind him. He improves the ship’s defenses. Ensnconced securely inside, he silently sets himself up as the ship’s alternate commander. He enlists the various operating functions of the ship to do his bidding, careful to avoid tripping any alarms. Captain Kirk is still up on the bridge in his swivel chair with the magnificent instrument arrays, unaware that he now has a rival in the depths of his ship. The Enterprise continues to perform as it always has. Meanwhile, the invader begins surreptitiously communicating with his own distant commander, letting him know that he is in position and ready, waiting for instructions.

And now imagine a vast fleet, in which the Enterprise is only one ship among millions, all of them
infiltrated in exactly the same way, each ship with its hidden pilot, ever alert to an outside command. In the real world, this infiltrated fleet is called a “botnet,” a network of infected, “robot” computers. The first job of a worm like Conficker is to infect and link together as many computers as possible—the phenomenon witnessed by Porras and other security geeks in their honeypots. Thousands of botnets exist, most of them relatively small—a few thousand or a few tens of thousands of infected computers. More than a billion computers are in use around the world, and by some estimates, a fourth of them have been surreptitiously linked to a botnet. But few botnets approach the size and menace of the one created by Conficker, which has stealthily linked between 6 million and 7 million computers.

Once created, botnets are valuable tools for criminal enterprise. Among other things, they can be used to efficiently distribute malware, to steal private information from otherwise secure Web sites or computers, to assist in fraudulent schemes, or to launch denial-of-service attacks—overwhelming a target computer with a flood of requests for response. The creator of an effective botnet, one with a wide range and the staying power to defeat security measures, can use it himself for one of the above scams, or he can sell or lease it to people who specialize in exploiting botnets. (Botnets can be bought or leased in underground markets online.)

Beyond criminal enterprise, botnets are also potentially dangerous weapons. If the right order were given, and all these computers worked together in one concerted effort, a botnet with that much computing power could crack many codes, break into and plunder just about any protected database in the world, and potentially hobble or almost any computer network, including those that make up a country’s vital modern infrastructure: systems that control banking, telephones, energy flow, air traffic, health-care information—even the Internet itself.

The key word there is could, because so far Conficker has done none of those things. It has been activated only once, to perform a relatively mundane spamming operation—enough to demonstrate that it is not benign. No one knows who created it. No one yet fully understands how it works. No one knows how to stop it or kill it. And no one even knows for sure why it exists.

If yours is one of the infected machines, you are like Captain Kirk, seemingly in full command of your ship, unaware that you have a hidden rival, or that you are part of this vast robot fleet. The worm inside your machine is not idle. It is stealthily running, issuing small maintenance commands, working to protect itself from being discovered and removed, biding its time, and periodically checking in with its command-and-control center. Conficker has taken over a large part of our digital world, and so far most people haven’t even noticed.

The struggle against this remarkable worm is a sort of chess match unfolding in the esoteric world of computer security. It pits the cleverest attackers in the world, the bad guys, against the cleverest defenders in the world, the good guys (who have been dubbed the “Conficker Cabal”). It has prompted the first truly concerted global effort to kill a computer virus, extraordinary feats of international cooperation, and the deployment of state-of-the-art decryption techniques—moves and countermoves at the highest level of programming. The good guys have gone to unprecedented lengths, and have had successes beyond anything they would have thought possible when they started. But a year and a half into the battle, here’s the bottom line:

The worm is winning.

A Digital Sam Spade

Twenty years ago, computers were bedeviled by hackers. These were savvy outlaws who used their deep knowledge of operating systems to invade, steal, and destroy, or sometimes just to tap into secure facilities and show off their skills. Hackers became heroes to a generation of teenagers, and had all sorts of motives, but their most distinctive trait was a tendency to show off.

Some had truly malicious intent. In his 1989 best seller, The Cuckoo’s Egg, Cliff Stoll told the story of
his stubborn, virtually single-handed hunt for an elusive hacker in Germany who was using Stoll’s computer system at the Lawrence Berkeley National Laboratory as a portal to Defense Department computers. For many people, Stoll’s book was the introduction to the netherworld of rarefied gamesmanship that defines computer security. Stoll’s hacker never penetrated the most secret corners of the national-security net, and even relatively serious breaches like the one Stoll described were more nuisance than threat. But the individual hacker working as a spy or vandal has evolved into something more organized and menacing.

Andre’ M. DiMino, a computer sleuth who is part of the Conficker Cabal, is considered one of the world’s foremost authorities on botnets. He stumbled into his avocation on a Monday morning a decade ago, when he discovered that over the weekend, someone had broken into the computer system he was administering for a small company in New Jersey. DiMino has an undergraduate degree in electrical engineering with an emphasis in computer science, but he has mostly taught himself up to his present level of expertise, which is extreme. At 45, he is a slender, affable idealist who keeps a small array of computers in an upstairs bedroom. When I stopped by to talk to him, he baked me pizza. His day job is doing computer forensics for law enforcement in Bergen County, New Jersey, but he has a kind of alter ego as what he calls a “botnet hunter.”

Back when he discovered the weekend break-in, DiMino assumed at first that it was the work of a hacker, a vandal, or possibly a former employee, only to discover, based on an analysis of the IP (Internet Protocol) addresses of the incoming data, that his little computer network had been invaded by someone from Turkey or Ukraine. What would someone halfway around the planet want with the computer system of a small business-management firm in a New Jersey office park? Apparently, judging by what he found, his invader was in the business of selling pirated software, movies, and music. Needing large amounts of digital storage space to hide stolen inventory, the culprit seemed to have conducted an automated search over the Internet, looking worldwide for vulnerable systems with large amounts of unused disc space—DiMino equates it to walking around rattling doorknobs, looking for one door left unlocked. DiMino’s system fit the bill, so the crooks had dumped a huge bloc of data onto his discs. He erased the stash and locked the door that had allowed the pirates in. As far as the company was concerned, that solved the problem. No harm done. No need to call the police or investigate further.

But DiMino was intrigued. He reviewed the server logs for previous weeks and saw that this successful invasion was one of many such efforts. Other attackers had been rattling the doors of his network, looking for vulnerabilities. If there were bad guys actively exploiting other people’s computers all over the world, designing sophisticated programs to exploit weaknesses ... how cool was that? And who was trying to stop them?

DiMino set about educating himself on the fine points of this obscure battle of wits. He eventually co-founded the Shadowserver Foundation, a nonprofit partnership of defense-minded geeks at war with malware, effectively transforming himself into a digital Sam Spade—indeed, the graphic atop Shadowserver’s home page features a Dashiell Hammett—style detective emerging from shadow.

Both sides in this cyberwar have become astonishingly sophisticated, operating at the cutting edge of programming theory and cryptography. Both understand the limits of security methodology, the one side working to broaden its reach, the other working to surpass it. Because malware has been automated, the good guys usually can only guess at who they are up against.

**Trojans, Viruses, and Worms**

Rodney Joffe heads the cabal that has been battling Conficker. He is a burly, garrulous South African–born American who serves as senior vice president and chief technologist for Neustar, a company that provides trunk-line service for competing cell-phone companies around the world. Joffe’s interest in stopping the worm did not stem just from his outrage and sense of justice. His
concern for Neustar’s operation is professional, and illustrative.

The company runs a huge local-number-portability database. Almost every phone call in North America, before it’s completed, must ask Neustar where to go. Back in the old days, when the phone company was a monopoly, telecommunications were relatively simple. You could figure out where a phone call was going, right down to the building where the target phone would ring, just by looking at the number. Today we have competing telephone companies, and cell phones, and a person’s telephone number is no longer necessarily tied to a geographic location. In this more complex world, someone needs to keep track of every single phone number, and know where to route calls so they end up in the right place. Neustar performs this service for telephone calls, and is one of many registries that oversee high-level Internet domains. It is, in Joffe’s words, “the map.”

“If I disappear, there’s no map,” he says. “So if you take us down, whole countries can actually disappear from the grid. They’re connected, but no one can find their way there, because the map’s disappeared.”

A botnet like Conficker could theoretically be used to shut down Neustar’s system. So Joffe helped form the Conficker Cabal. He scoffed when he read in late 2009 that the Obama administration’s Department of Homeland Security planned to hire “a thousand” computer-security experts over the next three years. “There aren’t more than a few hundred people in the world who understand this stuff.”

Most of us use the word virus to describe all malware, but in geekspeak, it means something more specific. There are three types of the stuff: Trojans, viruses, and worms. A Trojan is a piece of software that works like a Trojan horse, masquerading as one thing to get inside a computer, and then attacking. A virus attacks the host computer after slipping in through a hole in its operating system. It depends on the computer-operator—you—doing something stupid to activate it, like opening an attachment to an e-mail that appears innocuous, or clicking on an enticing link. A worm works like a virus, exploiting flaws in operating systems, but it doesn’t attack once it breaks in. It generally doesn’t have a malicious payload. Exactly like the most-sophisticated viruses in the biological world, it does not cripple or kill its host. It is primarily designed to spread. The instructions that will put a worm like Conficker to work are not embedded in its code; they will be delivered later, from a remote command center.

In the old days, when your computer got infected, it slowed down because your commands had to compete for processing with viral invaders. You knew something was wrong because the machine took 10 times longer to boot up, or there was a delay between command and response. You began to get annoying pop-ups on your screen directing you to download supposedly remedial software. Programs would freeze. In this sense, the old malware was like the Ebola virus, a very scary strain that messily kills nearly everyone it infects—which is another way of saying that it is grossly ineffective, because it burns out the very host organisms it needs to survive. The miscreants who created computer viruses years ago learned that malware that announces itself in these ways doesn’t last.

So today’s malware produces no pop-ups, no slowdowns. A worm is especially quiet, since all it does, at least initially, is spread. Conficker stealthily sets up shop without making a ripple, and—other than calling home periodically for instructions—just waits. Its regular messages to its command center amount to only a couple hundred bytes of data, which is not enough to even light up the little bulb that flashes when a computer hard drive is at work.

After Phil Porras and others began snaring Conficker in increasing numbers, they began dissecting it. The worm itself was exquisite. It consisted of only a few hundred lines of code, no more than 35 kilobytes—slightly smaller than a 2,000-word document. In comparison, the average home computer today has anywhere from 40 to 200 gigabytes of storage. Unless you were looking for it, unless you knew how to look for it, you would never see it. Conficker drifts in like a mote.
It exploited a specific hole, Port 445, in the Microsoft operating systems, a vulnerability that the manufacturer had tried to repair just weeks earlier. Ports are designated “listening” points in a system, designed to transmit and receive particular kinds of data. There are many of them, more than 65,000, because an operating system consists of layer upon layer of functions. A firewall is a security program that guards these ports, controlling the flow of data in and out. Some ports, like the one that handles e-mail, are heavily trafficked. Most are not; they listen for updates and instructions that deal with a narrow and specific function, usually routine procedures that never rise to the notice of computer-users. Only certain very specific kinds of data can flow through ports, and then only with the appropriate codes. Windows opens Port 445 by default to perform tasks like issuing instructions for print-sharing or file-sharing. Late in the summer of 2008, Microsoft learned that even a system protected by a firewall was vulnerable at Port 445 if print-sharing and file-sharing were enabled (which they were on many computers). In other words, even a well-protected computer had a hole. On October 23, 2008, the company issued a rare “critical security bulletin” (MS08-067) with a patch to repair that hole. A specially crafted “remote procedure call” could allow the port to be used by a remote operator, the security bulletin warned, and “an attacker could exploit this vulnerability without authentication to run arbitrary code.” The patch Microsoft offered theoretically slammed the door on a worm like Conficker almost a month before it appeared.

Theoretically.

In fact, the bulletin itself may have inspired the creation of Conficker. Many, many computer-operators worldwide—you know who you are—fail to diligently heed security updates. And the patches are issued only to computers with validated software installations; millions of computers run on bootlegged operating systems, which have never been validated. Microsoft issues its updates on the second Tuesday of every month. Every geek in the world knows this; it’s called “Patch Tuesday.” The company employs some of the best programmers in the world to stay one step ahead of the bad guys. If everyone applied the new patches promptly, Windows would be nigh impregnable. But because so many people fail to apply the patches promptly, and because so many machines run on illegitimate Windows systems, Patch Tuesday has become part of Microsoft’s problem. The company points out its own vulnerabilities, which is like a general responsible for defending a fort making a public announcement—“The back door to the supply shed in the southeast corner of the garrison has a broken lock; here’s how to fix it.” When there is only one fort, and it is well policed, the lock is fixed and the vulnerability disappears. But when you are defending millions of forts, and a goodly number of the people responsible for their security snooze right through Patch Tuesday, the security bulletin doesn’t just invite attack, it provides a map! Twenty-eight days after the MS08-067 security bulletin appeared, Conficker started worming its way into unpatched computers.

The Cabal’s Sandboxes

Conficker’s rate of replication got everyone’s attention, so a loose-knit gaggle of geeky “good guys,” including Porras, Joffe, and DiMino, began picking the worm apart. The online-security community consists of software manufacturers like Microsoft, companies like Symantec that sell security packages to computer owners, large telecommunication registries like Neustar and VeriSign, nonprofit research centers like SRI International, and botnet hunters like Shadowserver. In addition to maintaining honeypots, these security experts operate “sandboxes”—isolated computers (or, again, virtual computers inside larger ones) where they can place a piece of malware, turn it on, and watch it run. In other words, where they can play with it.

They all started playing with Conficker, comparing notes on what they found, and brainstorming ways to defeat it. That’s when someone dubbed the group the “Conficker Cabal,” and the name stuck, despite discomfort with the darker implications of the word. Here are some of the things the cabal discovered about the worm in those first few weeks:
• It patched the hole it came through at Port 445, making sure it would not have to compete with other worms. This was smart, because surely other hackers had seen security bulletin MS08-067.

• It tried to prevent communication with security providers (many computer-users subscribe to commercial services that regularly update antivirus software).

• When it started, if the IP address of the infected computer was Ukrainian, the worm self-destructed. When in attack mode, searching for other computers to infect, it skipped any with a Ukrainian IP address.

• It disabled the Windows "system restore" points, a useful tool that allows users with little expertise to simply reset an infected machine to a date prior to its infection. (System restore is one of the easiest ways to debug a machine.)

All of these things were clever. They indicated that Conficker's creator was up on all the latest tricks. But the main feature that intrigued the cabal was the way the worm called home. This is, of course, what worms designed to create botnets do. They settle in and periodically contact a command center to receive instructions. Botnet hunters like DiMino regularly wipe out whole malicious networks by deciphering the domain name of the command center and then getting it blocked. In the old days, this was easier because malware pointed to only a few IP addresses, which could be blocked by hosting providers and Internet service providers. The newer worms like Conficker bumped the game up to a higher level, generating domain names that involve many providers and a wide range of IP addresses, and that security experts can block only by contacting Internet registries—organizations that manage the domain registrations for their realm. But Conficker did not call home to a fixed address.

Shortly after it was discovered, the worm began performing a new operation: generating a list of domain names seemingly at random, 250 a day across five top-level domains (top-level domains are defined by the final letters in a Web address, such as .com or .edu or .uk). The worm would then go down the list until it hit upon the one connected to its remote controller's server. All Conficker's controller had to do was register one of the addresses, which can be done for a fee of about $10, and await the worm's regular calls. If he wished, he could issue instructions. It was as if the boss of a crime family told his henchmen to check in daily by turning to the bottom of a certain page in each day's Racing Form, where there would be a list of potential numbers. They would then call each number until the boss picked up. So it was not apparent from day to day where the worm would call home.

With the Racing Form trick, if you were a cop and were tipped off where to look, you might arrange with the paper's publisher to see the page before it was printed, and thus be one step ahead of the henchmen and their boss. To defeat Conficker, the geeks would have to figure out in advance what the numbers (or, in this case, domain names) would be, and then hustle to either buy up or contact every one, block it, or cajole whoever owned it to cooperate before the worm "made the call."

Michael Ligh, a young Brooklyn researcher employed by the computer-security company iDefense, is one of several people who went to work unraveling Conficker's methods. Ligh and others had seen algorithms for random-domain-name generation before, and most were keyed to the infected computer's clock. If new places to call home must be generated every day, or every few hours, then the worm needs to know when to perform the procedure. So the malware simply checks the time on its host computer. This provided the good guys with a tool to defeat it. They turned the clock forward on their sandbox computer, forcing their captured strain of the worm to spit out all the domain names it would generate for as long into the future as they cared to look. It was like stealing the teacher's edition of a classroom textbook, the one with all the answers to the quizzes and tests printed in the back. Once you knew all the places the malware would be calling, you could cordon off those sites in
advance, effectively stranding the worm.

Conficker had an answer for that. Instead of using the infected computer’s clock, the worm set its schedule by the time on popular corporate home pages, like Yahoo, Google, or Microsoft’s own msn.com.

“That was interesting,” Ligh said. “There was no way we could turn the clock forward on Google’s home page.”

So there was no easy way to predict the list of domain names in advance. But there was a way. The first step was to set up a proxy server to, in effect, intercept the time update from the big corporate Web site before it got back to the worm, alter the information, and then send it on. You could then tell the worm it was a date sometime in the future, and the worm would spit out the domain names for that date. This was a tedious way to proceed, since you could generate only one set of new domain names at a time. So Ligh and other researchers reverse-engineered the worm’s algorithm, extracted the time-update function, and wedded it to a piece of code they could control. They instructed their copy to generate the future lists in advance. They could then buy up or block all the sites, and direct all the worm’s communications into a “sinkhole,” a dead-end location where calls go unanswered. Conficker’s creators had deliberately made the task so onerous and expensive that no one would go to the trouble of blocking all possible command centers.

Or so they thought. The cabal, through a determined and unprecedented effort, did manage to cordon off the worm. By the end of 2008, Conficker had infected an estimated 1.5 million machines worldwide, but it was on its way to full containment. In the great chess match, the good guys had called “Check!”

Then the worm turned.

MD-6

On December 29, 2008, a new version of Conficker showed up, and if the geeks had been intrigued with the original version, they now experienced something more akin to respect ... mingled with fear.

One of the early theories about the worm was that it had slipped out of a computer-science lab, the product of some fooling-around by a sophisticated graduate student or group of students. They had loosed it on the world inadvertently, or maybe on purpose as a prank or experiment without realizing how effective it would be. This hypothesis appealed to optimists.

The new version of the worm, Conficker B, exploded the benevolent-accident theory. It was clear that the worm’s creator had been watching every move the good guys made, and was adjusting accordingly. He didn’t care that the good guys could predict its upcoming lists of domain names. He just rejiggered the worm to spread the new lists out over eight top-level domains instead of five, making the job of blocking them far more difficult. The worm had no trouble contacting all of these locations. If it received no command from one, it simply tried the next one on its list. Conficker B could go on like this for months, even years. It had to find its controller only once to receive instructions.

“That’s a high number,” Rodney Joffe, of Neustar, told me. “The cops will get sick and tired of knocking on 250 doors a day and finding there’s no one there. And if I’m the chief bad guy, all I have to do is be behind one of those doors on one of those days.”

There were other improvements to Conficker. Among them: besides shutting down whatever security system was installed on the computer it invaded, and preventing it from communicating with computer-security Web sites, it stopped the computer from connecting with Microsoft to perform Windows updates. So even though Microsoft was providing patches, the infected machines could not get to them. In addition, it modified the computer’s bandwidth settings to increase speed and propagate itself faster; and it began to spread itself in different ways, including via USB drives. This
last innovation meant that even “closed” computer networks, those with no connection to the Internet, were vulnerable, since users who cannot readily transmit files from point to point via the Web often store and transport them on small USB drives. If one of those USB drives, or a CD, was plugged into an infected computer, it could deliver the worm to an entire closed network.

All of this was impressive—but something else stopped researchers cold. Analysts with Conficker B isolated in their sandbox could watch it regularly call home and receive a return message. The exchange was in code, and not just any code.

Breaking codes used to be the province of clever puzzle masters, who during World War II devised encryption and code-breaking methods so difficult that operators needed machines to do the work. Computers today can perform so many calculations so fast that, theoretically at least, no cipher is too difficult to crack. One simply applies what computer scientists call “brute force”: trying every possible combination systematically until the secret is revealed. The game is to make a cipher so difficult that the amount of computing power needed to break it renders the effort pointless—the “thief” would have to spend more to obtain the prize than the prize is worth. In his 1999 history of code-making and -breaking, The Code Book, Simon Singh wrote: “It is now routine to encrypt a message [so securely] that all the computers on the planet would need longer than the age of the universe to break the cipher.”

The basis for the highest-level modern ciphers is a public-key encryption method invented in 1977 by three researchers at MIT: Ron Rivest (the primary author), Adi Shamir, and Leonard Adleman. In the more than 30 years since it was devised, the method has been improved several times. The National Institute of Standards and Technology sets the Federal Information Processing Standard, which defines the cryptography algorithms that government agencies must use to protect communications. Because it is the most sophisticated oversight effort of its kind, the standard is determined by an international competition among the world’s top cryptologists, with the winning entry becoming by default the worldwide standard. The current highest-level standard is labeled SHA-2 (Secure Hash Algorithm–2). Both this and the first SHA standard are versions of Rivest’s method. The international competition to upgrade SHA-2 has been under way for several years and is tentatively scheduled to conclude in 2013, at which point the new standard will become SHA-3.

Rivest’s proposal for the new standard, MD-6 (Message Digest–6), was submitted in the fall of 2008, about a month before Conficker first appeared, and began undergoing rigorous peer review—the very small community of high-level cryptographers worldwide began testing it for flaws.

Needless to say, this is a very arcane game. The entries are comprehensible to very few people. According to Rodney Joffe, “Unless you’re a subject-matter expert actively involved in crypto-algorithms, you didn’t even know that MD-6 existed. It wasn’t like it was put in The New York Times.”

So when the new version of Conficker appeared, and its new method of encrypting its communication employed MD-6, Rivest’s proposal for SHA-3, the cabal’s collective mind was blown.

“It was clear that these guys were not your average high-school kids or hackers or predominantly lazy,” Joffe told me. “They were making use of some very, very sophisticated techniques.

“Not only are we not dealing with amateurs, we are possibly dealing with people who are superior to all of our skills in crypto,” he said. “If there’s a surgeon out there who’s the world’s foremost expert on treating retinitis pigmentosa, he doesn’t do bunions. The guy who is the world expert on bunions—and, let’s say, bunions on the third digit of Anglo-American males between the ages of 35 and 40, that are different than anything else—he doesn’t do surgery for retinitis pigmentosa. The knowledge it took to employ Rivest’s proposal for SHA-3 demonstrated a similarly high level of specialization. We found an equivalent of three or four of those in the code—different parts of it.

“Take Windows,” he explained. “The understanding of Windows’ operating system, and how it worked
in the kernel, needed that kind of a domain expert, and they had that kind of ability there. And we realized as a community that we were not dealing with something normal. We’re dealing with one of two things: either we’re dealing with incredibly sophisticated cyber criminals, or we’re dealing with a group that was funded by a nation-state. Because this wasn’t the kind of team that you could just assemble by getting your five buddies who play Xbox 360 and saying, ‘Let’s all work together and see what we can do.’”

The plot thickened—it turned out that Rivest’s proposal, MD-6, had a flaw. Cryptologists in the competition had duly gone to work trying to crack the code, and one had succeeded. In early 2009, Rivest quietly withdrew his proposal, corrected it, and resubmitted it. This gave the cabal an opening. If the original Rivest proposal was flawed, then so was the encryption method for Conficker B. If they were able to eavesdrop on communications between Conficker and its mysterious controller, they might be able to figure out who he was, or who they were. How likely was it that the creator of Conficker would know about the flaw discovered in MD-6?

Once again, the good guys had the bad guys in check.

About six weeks later, another new version of the worm appeared.

It employed Rivest’s revised MD-6 proposal.

Game on.

“**Our Finest Hour**”

By early 2009, Conficker B had infected millions of machines. It had invaded the United Kingdom’s Defense Ministry. As CBS prepared a *60 Minutes* segment on the worm, its computers were struck. In both instances, security experts scrambled to uproot the invader, badly disrupting normal functioning of the system. Conficker now had the world’s attention. In February 2009, the cabal became more formal. Headed initially by a Microsoft program manager, and eventually by Joffe, it became the Conficker Working Group. Microsoft offered a $250,000 bounty for the arrest and conviction of the worm’s creators.

The newly named team went to work trying to corral Conficker B. Getting rid of it was out of the question. Even though they could scrub it from an infected computer, there was no way they could scrub it from all infected computers. The millions of machines in the botnet were spread all over the world, and most users of infected ones didn’t even know it. It was theoretically feasible to unleash a counter-worm, something to surreptitiously enter computers and take out Conficker, but in free countries, privacy laws frown on invading people’s home computers. Even if all the governments got together to allow a massive attack on Conficker—an unlikely event—the new version of the worm had new ways of evading the threat.

Conficker C appeared in March 2009, and in addition to being impressed by its very snazzy crypto, the Conficker Working Group noticed that the new worm’s code threatened to up the number of domain names generated every day to 50,000. The new version would begin generating that many domain names daily on April 1. At the same time, all computers infected with the old variants of Conficker that could be reached would be updated with this new strain. The move suggested that the bad guys behind Conficker understood not just cryptology, but also the mostly volunteer nature of the cabal.

“You know you’re dealing with someone who not only knows how botnets work, but who understands how the security community works,” Andre’ DiMino told me. “This is not just a bunch of organized criminals that, say, commission someone to write a botnet for them. They know the challenges that the security community faces internally, politically, and economically, and are exploiting them as well.”

The bad guys knew, for instance, that preregistering even 250 domain names a day at $10 a pop was doable for the good guys. As long as the number remained relatively small, the cabal could stay ahead
of them. But how could the good guys cope with a daily flood of 50,000? It would require an unprecedented degree of cooperation among competing security firms, software manufacturers, nonprofit organizations like Shadowserver, academics, and law enforcement.

“You can’t just register all 50,000—you’ve got to go one by one and make sure the domain name doesn’t already exist,” Joffe says. “And if it exists, you’ve got to make sure that it belongs to a good guy, not a bad guy. You’ve got to make a damn phone call for any of the new ones, and have to send someone out there to do it—and these are spread all over the world, including some very remote places, Third World countries. Now the bar had been raised to a level that was almost insurmountable.”

The worm was already running rings around the good guys, and then, just for good measure, it planted a pie in their faces on, of all days, April 1. By playing with the new variant in their sandboxes, the cabal knew that the enhanced domain-name-generating algorithm would click in on that day. If the update succeeded, it would be a game-changer. It was the most dramatic moment since Conficker had surfaced the previous November. Apparently, at long last, this extraordinary tool was going to be put to use. But for what? The potential was scary. Few people outside the upper echelon of computer security even understood what Conficker was, much less what was at stake on April 1, but word of a vague impending digital doomsday spread. The popular press got hold of it. There were headlines and the usual spate of ill-informed reports on cable TV and the Internet. When the day arrived, those who had been warning about the dangers of this new worm were sure to see their fears vindicated.

The cabal mounted a heroic effort to shut down the worm’s potential command centers in advance of the update, coordinating directly with the Internet Corporation for Assigned Names and Numbers, the organization that supervises registries worldwide. “It was our finest hour,” Joffe says.

“I don’t think that the bad guys could have expected the research community to come together as it did, because it was pretty unprecedented,” Ramses Martinez, director of information security for VeriSign, told me. “That was a new thing that happened. I mean, if you would have told me everybody’s going to come together—by everybody, I mean all these guys in this computer-security world that know each other—and they’re going to do this thing, I would have said, ‘You’re crazy.’ I don’t think the bad guys could have expected that.”

Much of the computer world was watching, in considerable suspense, to see what would happen on April 1. It was like the moment in a movie when the bad guy at last has cornered the hero. He pulls out an enormous gun and aims it at the hero’s head, pulls the trigger ... and out pops a little flag with the word BANG!

Conficker found one or two domain names that Joffe’s group had missed, which was all it needed. The cabal’s efforts had succeeded in vastly reducing the number of machines that got the update, but the ones that did went to work distributing a very conventional, well-known malware called Waledac, which sends out e-mail spam selling a fake anti-spyware program. The worm was used to distribute Waledac for two weeks, and then stopped.

But something much more important had happened. The updated worm didn’t just up the ante by generating 50,000 domain names daily; it effectively moved the game out of the cabal’s reach.

“April 1 came and went, and in the middle of that night the systems switched over to the new algorithm,” Conficker C, Joffe told me. “That’s all that was supposed to happen, and it happened. But the Internet didn’t get infected; it was just an algorithm change in the software. So of course the press said, ‘Conficker is a bust.’”

Public concern over the worm fizzled, just as the problem grew worse: the new version of Conficker introduced peer-to-peer communications, which was disheartening to the good guys, to say the least. Peer-to-peer operations meant the worm no longer had to sneak in through Windows Port 445 or a
USB drive; an infected computer spread the worm directly to every machine it interacted with. It also meant that Conficker no longer needed to call out to a command center for instructions; they could be distributed directly, computer to computer. And since the worm no longer needed to call home, there was no longer any way to tell how many computers were infected.

In the great chess match, the worm had just pronounced “Checkmate.”

Watching and Waiting

As of this writing, 17 months after it appeared and about a year after the April 1 update, Conficker has created a stable botnet. It consists of anywhere from hundreds of thousands of computers to 12 million. No one knows for sure anymore, because with peer-to-peer communications, the worm no longer needs to check in with an outside command center, which is how the good guys kept count. Joffe estimates that with the four distinct strains (yet another one appeared on April 8, 2009), 6.5 million computers are probably infected.

The investigators see no immediate chance or even any effective way to kill it.

“There are a bunch of infected machines that are out there, and they can be taken over, given the right circumstances, by the bad guys,” VeriSign’s Martinez says. “Will they do that? I don’t know. So it’s a potential threat. It’s something that’s out there, sitting there, and it needs to be addressed, but I don’t think, honestly, that we know how. How do we address this? If it was sitting in the U.S., it would be a fairly easy thing to do. The fact is that it’s spread out all around the world.”

Ever since the paltry Waledac scam, the worm has been biding its time.

“They are watching us watch them,” says Andre’ DiMino, the botnet hunter. “I think it’s really either that or somebody let this thing get bigger, and it’s advanced bigger and further than they ever dreamed possible. A lot of people think that. But in looking at the sophistication of this thing and looking at the evolution of this thing, I think they knew exactly what they were doing. I think they were trying something, and I think that they’re too smart to do what everybody figured they were going to do. You have to remember, the world was watching this thing and waiting for the world to end from Conficker on April 1, 2009. The last thing you’d want to do if you’re the bad guy is make something happen on April 1. You’re never going to do that, because everybody’s watching it. You’re going to do something when you’re least suspected. So these guys are sophisticated. They have good code. And just even seeing the evolution from Conficker A to B to C, where there’s the peer-to-peer component, which ... strikes fear into the heart of botnet hunters because it’s just so damn difficult to track—these guys know exactly what they’re doing.”

SO WHO ARE they?

One of the things Martinez’s team does, patrolling the perimeter at VeriSign looking for threats, is dip into the obscure digital forums where cyber criminals converse. Those who are engaged in writing sophisticated malware boast and threaten and compare notes. The good guys venture in to collect intelligence, or just out of curiosity, or for fun. They sometimes pretend to be malware creators themselves, sometimes not. Sometimes they engage in a little cyber trash talk.

“In the past you were just sort of making sure they didn’t steal your proprietary information,” Martinez says. “Now we go in to engage them. You talk to them and you exchange information. You have a guy in Russia selling malware, working with a guy in Mexico doing phishing attacks, who’s talking to a kid in Brazil, who’s doing credit-card fraud, and they’re introducing each other to some guy in China doing something else.”

Martinez said he recently eavesdropped on a dialogue between a security researcher and a man he suspects was at least partly responsible for Conficker. He wouldn’t say how he drew that connection, only that he had good reasons for believing it to be true. The suspect in the conversation was eastern
European. The standard image of a malware creator is the Hollywood one: a brilliant 20-something with long hair and a bad attitude, in need of a bath. This is not how Martinez sees his nemesis—or nemeses.

“I see him, or them, as a really well-educated, smart businessman,” he said. “He may be 50 years old. These guys are not chumps. They’re not just out to make a buck.”

The eastern European, backpedaling from further dialogue with the security geek, wrote, “You’re the good guys; we’re the bad guys. Bacillus can’t live with antibodies.”

“Now, I didn’t grow up in a bad neighborhood or anything,” said Martinez, “but the few thugs that I saw would never use a word like bacillus or make an analogy like that.”

One of the early clues in the hunt was the peculiarity in the Conficker code that made computers with active Ukrainian keyboards immune. Much of the world’s aggressive malware comes from eastern Europe, where there are high levels of education and technical expertise, and also thriving organized criminal gangs. Martinez believes Conficker was written by a group of highly skilled programmers. Like Joffe, he sees it as a group of creators, because designing the worm required expertise in so many different disciplines. He suspects that these skilled programmers and technicians either were hired by a criminal gang, or created the worm as their own illicit business venture. If that’s true, then the Waledac maneuver was like flexing Conficker’s pinkie—just a demonstration, a way of showing that despite the best and most concerted effort of the world’s computer-security establishment, the worm was fully operational and under their control.

Will they be caught?

“I have no idea,” Martinez says. “I would say probably not. I’ll be shocked if they’re ever arrested. And arrest them for what? Is breaking into people’s computers even illegal where they’re from? Because in a lot of countries, it isn’t. As a matter of fact, in some countries, unless you’re touching a computer in their jurisdiction, their country, that’s not illegal. So who’s going to arrest them, even if we know who they are?”

Ridding computers of the worm poses another kind of overwhelming problem.

“There are controls, or checks and balances, in place to limit what police can do, because we have civil liberties to protect,” he says. “If you do away with these checks and balances, where the government can come in and reimagine your computer overnight, now you’re infringing on people’s civil liberties. So, I mean, we can talk about this all day, but I’ll tell you, it’s going to be a long time, in my opinion, before we really see the government being able to effectively deal with cyber crime, because I think we’re still learning as a culture, as a nation, and as a world how to deal with this stuff. It’s too new.”

Imagining Conficker’s creators as a skilled group of illicit cyber entrepreneurs remains the prevailing theory. Some of the good guys feel that the worm will never be used again. They argue that it has become too notorious, too visible, to be useful. Its creators have learned how to whip computer-security systems worldwide, and will now use that knowledge to craft an even stealthier worm, and perhaps sell it to the highest bidder. Few believe Conficker itself is the work of any one nation, because other than the initial quirk of the Ukrainian-keyboard exemption, it spreads indiscriminately. China is the nation most often suspected in cyber attacks, but there may be more Conficker-infected computers in China than anywhere else. Besides, a nation seeking to create a botnet weapon is unlikely to create one as brazen as Conficker, which from the start has exhibited a thumb-in-your-eye, catch-me-if-you-can personality. It is hard to imagine Conficker’s creators not enjoying the high level of cyber gamesmanship. The good guys certainly have.

“It’s cops and robbers, so to speak, and that was a really interesting aspect of the work for me,” says Martinez. “It’s guys trying to outwit each other and exploit vulnerabilities in this vast network.”
In chess, when your opponent checkmates you, you have no recourse. You concede and shake the victor’s hand. In the real-world chess match over Conficker, the good guys have another recourse. They can, in effect, upend the board and go after the bad guys physically. Which is where things stand. The hunt for the mastermind (or masterminds) behind the worm is ongoing.

“It’s an active investigation,” Joffe says. “That’s all I can say. Law enforcement is fully engaged. We have some leads. This story is not over.”

This article available online at:


Copyright © 2010 by The Atlantic Monthly Group. All Rights Reserved.