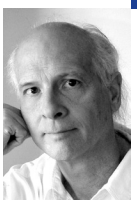
RIK FARROW

musings



Rik is the Editor of *;login:*.

rik@usenix.org

ANOTHER DAY, ANOTHER EXPLOIT. Today I read that some researchers plan on demoing a rootkit for Android at DefCon 2010. In May, UW and UCSD researchers announced remote control of a car via the diagnostic port [1] and a wireless link. They could disable the brakes totally, apply the brakes to just one wheel (or more), stall the engine, and control the display panel. And just wait until you learn what can be done with your new smartphone.

I got interested in computer security because I found it both intriguing and challenging. Understanding security involves knowledge of programming, system administration, OS design and implementation, and networking. Few CS disciplines cross so many boundaries. I've done well in the field, but still feel like I am groping in the dark.

Visualization

Speaking of darkness, I decided to visualize what government-sponsored attackers might be using in the near future. It would have been really nice to have a console that was not only used for attacking (or, more politely, penetration testing), like Metasploit, but something that displays the status of the systems and networks that may be involved in the attack. Of course, I also imagine that the operator can call up lists of potential attacks and chose to execute an attack with a few keystrokes (real hackers keep their hands on the keyboard—mouses are for sissies). On that note, here goes . . .

Maxim glanced at his smartphone, checked the time, and stubbed out his cigarette. He really hated himself for smoking, but did it for the sharpened focus it seemed to bring. He looked around the stuffy, windowless room, with its piles of books, magazines, and stacks of paper, then turned out the dim LED lamp over his desk. Finally, he lifted the helmet from its rest and settled it into position.

Maxim sighed deeply. As usual, he felt enthralled by the display. He appeared to be floating in the depths of space, but instead of stars, collections of brightly colored lights surrounded him. With a couple of keystrokes, he could zoom in on an area and view the tags that provided more detail. Or he could add in overlays that showed the amount of network traffic between any of the targets in his display.

Soft pops, crackles, and chimes came from all directions. The UI people had decided that enhanc-

ing the display with sound was a good idea, and Maxim agreed. Humans retain ancient instincts from their hunter-gatherer days, when someone who ignored a cracking branch might miss finding dinner, or quickly become dinner. As new sources of information appeared, they chimed softly, then crackled as more information about the source was added.

What Maxim really hated was the thumper. If the back-end AI decided that the attacker had been detected, it not only produced a tremendous *crump* sound, it also thumped the seat hard enough to jolt his whole body. Full surround experience. Fortunately, he hadn't been detected often since his training days.

Maxim had discovered many years ago that he worked best when he could visualize what he was working with. In junior high, he had scribbled out graphical representations for the systems he was attacking, and how they appeared to be related. His drawings were crude, and trying to decipher scrawled IP addresses and port numbers later always proved troublesome. Plus, trying to recall what he had written to represent passwords was truly frustrating.

By comparison, this new visualization system was a dream come true. Just selecting the object representing a target brought up all the known details, as well as some that were guesses, and allowed him to scroll through lists of actions he could use to probe or attack.

Maxim watched as the display continued to evolve, with new objects glowing and chiming into life in front of him, while others faded away as they moved behind him in the virtual display. He still had a minute or so before the action began, and he allowed himself to daydream. How would things have looked in this display when he first started hacking back in the late '90s?

Instead of objects and their attributes, Maxim conjured up an image made of dots of light in the darkness. Each dot represented an open port, and the text associated with the dot gave the port number and, for a well-known port, perhaps the name of the protocol. Port scans produced the raw data, and seeing it as dots of light made quick assessments of targets easy. A totally black display meant no ports open, while one full of glowing dots suggested a wide-open, poorly configured system just wanting to be owned. Sometimes such systems would already have been exploited many times before he found them. Occasionally, he would be the first.

Times changed, and people became more aware of security. Firewalls became common, so instead of scanning target systems, all you could scan was the firewall. And scanning firewalls was akin to throwing rocks at an armed guard sitting inside a bunker—unless the firewall was poorly configured. Sometimes a port scan of a firewall would produce a display every bit as useful as a wide-open server. Other times, the firewall would stop responding to his scan, leaving nothing but darkness in its place.

Where once the targets had mostly been various UNIX and Linux servers, Windows machines started appearing. Recognizing them from the open ports was as easy as recognizing an old Ford by its square tail lights. UNIX systems, long the victim of attacks, were still common but were much better secured these days. No more easy shell accounts, but Windows systems were now sitting on faster networks, making them useful as relays and exploit stashes.

Maxim mentally tagged each machine he had exploited with a different color: red for firewalls, blue for relays, and orange for code stashes. Over time, the display would change as exploited machines were taken down,

patched, or wiped clean. Often he could re-exploit a system, but he would always wait a few weeks before doing so.

Maxim labeled Web servers as green, because, for a while, they had meant money. Web servers always had open ports because serving data was their job. A firewall could still get in the way, but few people bothered to configure them correctly, something that would have made downloading his tools and attacks more difficult. With a Web server as a relay, he could usually get inside an organization in a snap.

Then the Web servers themselves became sources of income. Maxim would pop Web servers, add a single line to many HTML files that would direct any browser that downloaded them to a drive-by download site, and collect money for each system exploited. It was a great time, like being a highwayman, or even a troll, the storybook kind that lived under bridges.

He still took side jobs, and these were a little more demanding. In most cases, someone wanted to get something from within a particular organization. If his clients provided him with lots of specifics, such as the names and email addresses of a target and his associates, he could use a classic social engineering attack. The target would get an email and by opening the email, silently exploit his Windows desktop. Visualizing this, Maxim imagined a long tunnel stretching from his desktop, through any firewalls, and right into the target organization. Once inside, he would slowly build a picture of the internal network by cautious probing (don't want to set off an IDS by being too eager) and reading emails. Just the headers alone would sometimes be enough to steer him in the right direction.

Maxim popped back into the present moment. His fingers floated ready above his keyboard, waiting to launch the next phase of the attack. His display had become almost muted, as the object he was focused upon moved into a secure corridor, one with little network traffic. After a brief pause, the object moved forward and the screen lit up again. Bingo! He was inside. Maxim's hands flashed as he typed out macros, launching several attacks in swift succession. It was important to establish a beachhead, by setting up relays, before the subject turned off his smartphone. Until then, the smartphone acted as a relay between a tunnel to his attacking system and the wireless network the smartphone had joined. He didn't know, nor did he need to know, the encryption key for the wireless, since the smartphone had handled that for him. And with data rates in metro areas approaching five megabits per second, any downloads he needed to do would go fast.

Maxim perceived the richness of the display grow as the passive sniffing tool he had launched did its work. A print server appeared, and Maxim played a few keys that started a sure-fire exploit against the server. Print servers were often trusted in office networks, and were a great place to "share" malware.

It really wasn't hard to find services on today's networks, what with all the announcements. Routers, printers, file servers, authentication servers, all either routinely broadcast or responded to broadcast queries for attention.

Maxim already knew who would likely hold the key to the data file his employers wanted. All he needed to do was discover the IP address assigned to that user's desktop. There! A plaintext email header with the name in the To: header line gave away the address. Maxim launched the exploit that would do the deed. By sending out a fake ARP reply, the smartphone would relay data between the victim and the default router. Once the victim visited a Web site, the relay would insert a bit of JavaScript code that, once executed, would add a relay to the victim's Web browser. Maxim loved IE6, as IE8 would have made this part of the attack much more difficult.

The sound of a gong, like one from an Asian monastery, announced success. Maxim quickly shut down the ARP spoofing program, as it would be easily detected by IDS (and perhaps already had been). He then wiped his tools from the smartphone and closed the tunnel. Instantly the display went dark as the connection closed, accompanied by the pops that signified systems disappearing.

Maxim took a deep breath, coughed, and removed the helmet. He glanced at his phone as he lit a cigarette: he had been inside for just five minutes.

Fantasy to Reality

Maxim's visualization tool is currently, AFAIK, total fantasy. The attacks I described are not fantasy. The past attacks are part of history. Using a smartphone as a relay is the only new attack presented, and it is technically possible today. On the best secured phones, such as the maniacally controlled iPhone, this attack would be difficult and would rely on jailbreaking. But on phones where users can install any app they choose and grant the app all the permissions it asks for, this attack becomes trivial. The user might notice how sluggish his phone had become but would be unlikely to notice anything else. The entire display of a smartphone is driven by software and can be convinced to display anything at all, easily fooling the user into thinking things are perfectly normal. The display could even fake a complete shutdown, as the only way you can really be sure a smartphone is off is to remove the battery—something you can't do routinely with an iPhone.

I disliked using computers when they were so complex that no single person could understand the entire system. The mainframes I used in the '70s were like that, similar to mainframes that are still in use today. These mainframes required many bookshelves to hold the documentation for system programmers, those allowed to hook into the underlying OS.

When I started using PCs, understanding the entire system was much easier. After all, a system with 64kb of RAM and only floppy disks just couldn't hold very much, and having a complete understanding of how the system worked was possible. Early UNIX systems were the same way, with a total of three binders containing *all* of the documentation. Today's *nix systems are no longer so simple—a kernel alone uses tens of thousands of times the RAM my entire first PC had.

Layered on top of any modern desktop or server, we find libraries. The libraries make life much easier for programmers, as they provide consistent and easy-to-use interfaces with the rest of the system. They also add many layers between the hundreds (for *nix) or thousands (for Windows) of system calls. Each layer of abstraction adds complexity even as it reduces the programmer's burden, while opening up more opportunities for exploitation.

My friends in the business of AV and application auditing have been telling me how much better Windows security has become. They swear Windows 7 has much better security than either Linux or Mac OS X because of many great new features. Yet there are still exploits for Windows 7. The same people tell me that this is because Windows versions control 95% of desktops, and that is certainly true to a point. But as long as there are many layers of software, and users are allowed to install any software they like or run browsers that will do this for them, Windows will not be secure.

And neither will other systems.

The day of the user-controlled system may be coming to an end. For security, this will be a good thing. For freedom, including the freedom to tinker,

it might be terrible. Yet Apple has already done this with its iP* family, and I wonder if Microsoft Windows and Google Android will be far behind.

The Lineup

This issue begins with an article about logging, not in the sense of mining the land for timber, but mining your logs for the interesting parts. Ron Dilley has been obsessed with logs for many years. He tells us how this obsession has lead him to create a set of useful tools and techniques for monitoring systems.

Steve Checkoway and friends have written about some research they published at LEET '10 (reports in this issue). Checkoway warns us that even text isn't safe, and demonstrated this by creating a virus that infects systems via TeX files.

Bo Li et al. explain the Symbian OS security model. Li begins with some background on the original design, then covers the current security features used in the most popular smartphone OS in the world today (based on number of handsets sold). I find that the trade-offs between security and the desire to have lots of apps for your OS tends to favor apps over security—but I suggest you find out for yourself.

Carlos Maltzahn and associates have provided us with an overview of Ceph, a distributed file system in the works for over five years, and they explain how to integrate Ceph with Hadoop. In the April 2010 issue of *;login:*, Konstantin Shvachko discussed how the single namespace server design of HDFS creates very real limits in scaling and performance. Ceph bypasses these limitations by having many metadata servers. The client that allows mounting of Ceph was added to the Linux kernel in May, making Ceph even more accessible. The instructions in this article for setting up your own Ceph distributed file system are clear and simple to follow.

Michael Piatek and Arvind Krishnamurthy explain why the tit-for-tat reward system used in peer-to-peer file-sharing systems like BitTorrent can't work with live video streaming. In a short and very clear article based on their paper for NSDI, they outline a different strategy, one that has already been implemented in a commercial video streaming product.

David Blank-Edelman expands on the theme of his April column, making things up, by covering randomness. If you have been awake and interested in computer security, you will appreciate just how important good sources of randomness can be. David takes us on a tour of Perl modules that go beyond the two system calls POSIX operating systems provide.

Peter Galvin stares into his crystal ball, then chucks it. Peter takes a hard look at the future of various Sun products under Oracle control, basing his column on the few facts that are known and a bit of deduction. I recommend his column for anyone who is interested in the future of Sun.

Dave Josephsen continues where he left off last month in his discussion of Argus. In this column Dave expounds on the basics of collecting packet summary data using Argus, explains how to combine, collate, and reduce Argus data, and demos some commands for extracting interesting stuff from this data.

Robert Ferrell provides a hard look at disaster planning. Possibly inspired by the opening of the hurricane season, with all the destruction, death, and misery that implies, Robert suggests that the best disasters are the ones you plan yourself.

We have reports from the NSDI conference and workshops in this issue. All of the workshops were covered. We also have a report from BSDCan 2010, about an invited talk that considers the interesting side effects you can expect from implementing IPv6.

I just burned a new Linux live-CD for a friend, one that he uses only for online banking. Windows rootkits have become less common: exploiting the browser is so much easier, why bother installing a rootkit? Not that rootkit-like malware does not still exist, and is in fact pretty common in bot clients. By booting from a CD, my friend has a guaranteed secure path from his keyboard, through the browser and its many layers of libraries, to his banking site (which may or may not be secure).

If I had my way, all systems would provide as much security as booting from a live-CD can. I think it is at least possible that we will see such systems soon enough, as Android strives to produce this level of security, and the iPhone already comes close. But that day is still in the future. Be very careful, and perhaps you can avoid the messiness of being exploited.

REFERENCE

[1] Karl Koscher et al., "Experimental Security Analysis of a Modern Automobile," *2010 IEEE Symposium on Security and Privacy*: http://www.autosec.org/pubs/cars-oakland2010.pdf.