Defcon Capture the Flag: Defending Vulnerable Code from Intense Attack

Crispin Cowan, Seth Arnold, Steve Beattie, and Chris Wright

WireX Communications, Inc. http://wirex.com/

John Viega

Secure Software Inc. http://securesoftware.com/

Abstract

Defcon's Capture the Flag (CtF) game is the largest open computer security hacking game. This year's CtF hat rules that made it particularly difficult to be a successful defender. We entered an Immunix server, comprised of five years of IA&S, OASIS, FTN, and CHATS technologies, to see whether this system could survive sustained attack from determined experts. We describe our experience surviving Defcon CtF.

1 Introduction

Defcon bills itself as "the largest underground Internet security gathering on the planet." Defcon also provides a "Capture the Flag" (CtF) contest; a weekend-long contest of computer security attack and defense skills. As a large-scale *legal* opportunity for attackers to demonstrate their "eleet skilz" in a public forum, Defcon CtF attracts a large pool of talented attackers. We entered an Immunix server into this contest to test the efficacy of five years of accumulated DARPA survivability R&D against this rich pool of adversaries.

"Capture the Flag" is a poor metaphor for the structure of the Defcon CtF game, as there is no single flag to capture. Rather, each team has to defend its own flag, while trying to corrupt the flags of as many of the other teams as possible as shown in Figure 1. A "flag" is a data file on a server; each flag identifies a team. Initially, each team's server has their own flag on it. Throughout the game, attackers seek to replace the flag on someone else's server with their own flag, while defenders try to preserve their own flag on their own server. A score server periodically polls the player servers to detect the identity of the flag on each, and score the game accordingly.

It should be noted that, in contrast to traditional DARPA "red teaming" exercises, this game is *symmetric*, in that each team has both attackers and defenders. *Asymmetric* red team exercises have the advantage of allowing the modeling of asymmetric threats, reflecting the asymmetric threat in the real world. However, asymmetric testing has led to problems with interpretation of the rules of engagement, leading to disputes about whether a given attack was "fair" or "within scope." Symmetric gaming obviates the rules of engagement problem, because all players are subject to the same rules.

Another effect of symmetric gaming is that all teams must have attackers, which would be problematic if symmetric gaming were adopted to evaluate DARPA defensive technologies, because not all defensive technology developers have attack skills. The solution to this problem is to take whatever attacker resources are available, and apportion them among defending teams.[1]

The rest of this paper is organized as follows. Section 2 describes the details of the 2002 CtF game, including major upgrades from recent years. Section 3 describes the Immunix server that we entered in the game. Section 4 recounts the play-by-play action as the game progressed. Section 5 presents strategic analysis of how we approached the game. Section 6 describes tactical analysis of attacks deployed against us, and attacks we deployed. Section 7 outlines future work. Section 8 presents our conclusions. Section 9 presents acknowledg-

1. It should be noted that WireX does not have significant attack capabilities. Attack capabilities were provided by recruiting friends & associates, as well as *ad hoc* interested parties at the conference. Our co-author John Viega was instrumental in bringing significant attack capabilities to our team. See Section 9 for the full list of acknowledgments.

This work supported in part by DARPA contract N66001-00-C-8032.

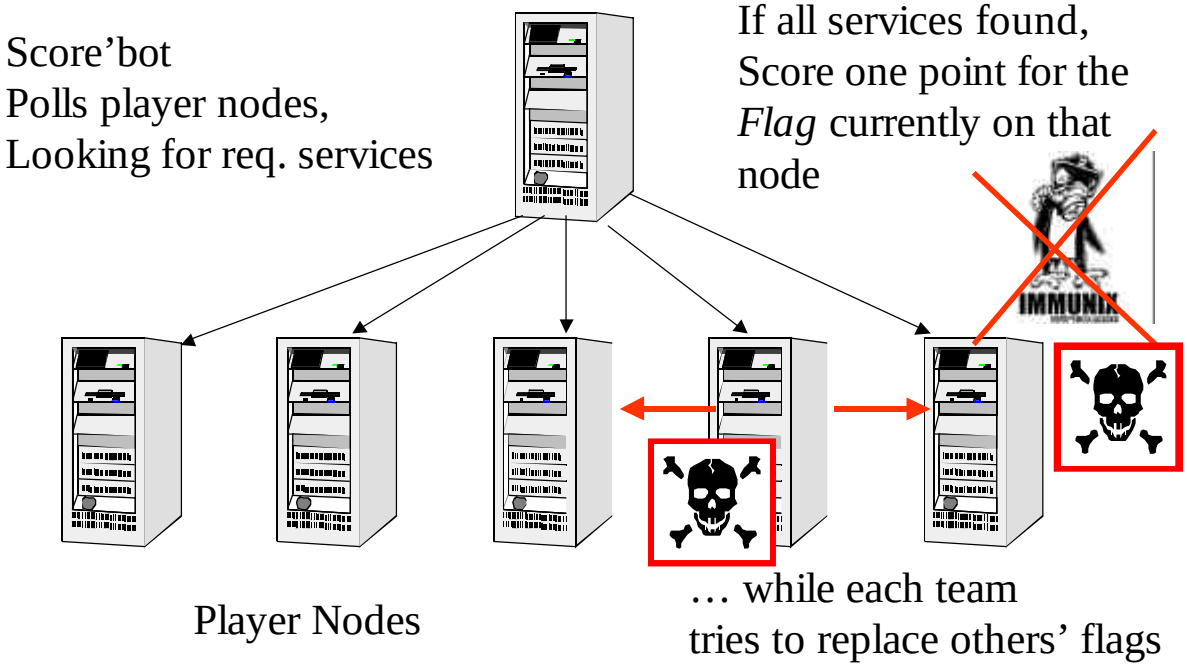

Figure 1 Defcon Capture the Flag (CtF)

ment to the very large group of people who helped with this effort.

2 The Contest

In recent years, the Defcon CtF game had ossified: teams were permitted to bring well-prepared servers to the game, with no specific requirements as to the functionality that the server must provide. The result was very low scoring games, in which few successful penetrations occurred, resulting in boring, low-scoring "soccer matches." The game was further undermined by subjective judgements like "style points" for particularly creative ways to penetrate systems. To address this problem, a new team of game organizers called the "Ghettohackers" [13] designed an improved CtF game to provide a more interesting challenge to the players, and a more interesting spectacle to the audience.

The new game attempts to test a security administrator's ability to secure a complex system with unknown-but-required functionality. While this task seems rather odd, the Ghettohackers defend it as being similar to a member's day job as a consultant: a customer has a large dot.com site, they don't know what it does (the IT staff have all left), and they want it to be secure. And don't turn it off, there is live traffic running on it. The Ghettohackers CtF game models this situation as follows:

- Players are provided with a table, one power outlet, and one ethernet connection.
- Players get a class-C network address space, and all traffic coming to the player's connection is reverse-NAT'd so that the source of traffic cannot be identified. This eliminates the obvious defense of filtering all traffic from other teams using a simple firewall.
- Players are handed a reference system at the beginning of the game. The reference system is guaranteed to provide all the services required by the score server.
- The actual services required by the score server are *secret*, and subject to change throughout game play.
- The reference system is riddled with security vulnerabilities, and (as it turned out) included inherently insecure services, such as telnet and FTP.
- To score a "home" point, a team's server must fully satisfy the score server's requested interactions, and the team's flag must be intact on their server.
- To score an "own3d" point, the score server must be fully satisfied with the services on *other* team's server, the attacking team's flag must be present on other team's server, *and* the attacking team's server must also be fully functional. This is to prevent a team from deploying *only* attackers, and not bothering to defend.

Figure 2 Nasdaq-like Score Board

- To discourage DoS attacks and lazy bulk scanning, each team is charged a penalty for bandwidth coming from their connection.

Successfully defending such a system from attack, while simultaneously providing the required services to the score server, is a difficult challenge, resulting in a much more high-scoring game with more action: servers are frequently penetrated. In addition to making the game more interesting, the difficulty also serves to compress longer-term real-world security effects into a weekend of game play. Apart from the security challenge, a major problem presented by the secrecy of the score server's requirements is to *intuitively guess* which traffic appearing at your server are requests from the score server, and which are attacks.

This year's game also included a significant upgrade to the score display. A large, colorful display modeled after a stock market ticker was used to indicate how each team was doing *lately*, shown in Figure 2. This had several effects:

Obfuscate actual performance: The display did not directly indicate how each team was doing in *total*, only a vague relative performance indication. This was intentional, so as to not discourage teams from continuing to play, by not telling them how far behind they might be.

Entertain the audience: The display was very effective at keeping the audience (convention attendees) engaged in the progress of the game. The display was accompanied by music and punctuated by videos and announcements with a "Blade Runner" futuristic theme.

Overall, the Ghettohackers were successful. The CtF game was widely praised by conference attendees as

being a large step forward over previous years. The game room was crowded, with many conference attendees staying to watch as the game unfolded. The game also garnered good press, both on TechTV and on the web [14]. In Section 4 we recount the progress of the game.

3 Our Entry: Immunix

Our entry in the CtF game was an Immunix server. Immunix is a security-hardened version of Linux, protected with the following technologies:

StackGuard: A C compiler enhancement [8] that emits programs resistant to buffer overflow attacks [16, 9]. This technology was developed under DARPA contracts F30602-96-1-0331 (Immunix), F30602-96-1-0302 (Heterodyne), and F30602-01-C-0172 (Sardonix).

FormatGuard: A similar C compilation technique [4] that emits programs resistant to printf format string vulnerabilities [18, 2, 15]. This technology was developed under DARPA contract N66001-00-C-8032 (Autonomix).

RaceGuard: A kernel enhancement [6] to detect and stop temporary file race attacks [1]. This technology was developed under DARPA contract N66001-00-C-8032 (Autonomix).

SubDomain: A mandatory access control scheme [5] that lets the kernel enforce the set of files that can be accessed by each *program*. This technology was developed under DARPA contract F30602-96-1-0331 (Immunix) and commercially by WireX.

Openwall: A kernel enhancement to make the stack segment of program address spaces non-executable [10]. This technology is a popular open source result from Russia.

The Immunix system, protected with these technologies, offers a reasonably high degree of security, and a high degree of compatibility with standard Red Hat Linux systems. Our goals in entering Immunix in the Defcon CtF were to show that Immunix was secure enough to survive concerted attack from numerous expert attackers, and show that it is feasible to rapidly port software onto Immunix and expect it to work.

4 Play by Play

At 11 am Friday morning, all team captains were handed a CD with the reference system on it, and given an hour or so to get their server up before the score server's first poll. The reference system was actually a VMWare 3.0 image. There were teething problems with many teams involving VMWare, resulting in delays starting the game, and play eventually started around 2 pm Friday.

The system itself was a modified Red Hat Linux 6.2, *not* patched for assorted vulnerabilities, and provisioned with Apache running as root and some "interesting" CGIs which allowed anonymous CGI users to add and delete arbitrary users, with arbitrary user-IDs including zero (root). Nmap [12] of the reference server showed nearly every common port open, and several uncommon ones.

4.1 Friday: Configuration Problems

The popular strategy was to launch the reference system, and use *ad hoc* human intrusion detection to detect & halt intrusions, and patch things up as best as possible.

The Immunix strategy was to inspect the reference system, and port the services to the pre-configured Immunix server we brought to the game as quickly as possible. In the first four hours of the game, our server was down while we enabled what we guessed were required network services and put SubDomain profiles around them. By 6 pm, we had our Immunix server in a state where we were at least confident that it would not be compromised, and launched it. Unfortunately, it took an additional six hours of work to refine the services we offered to the point where the score server was satisfied with our services.

The difficulty was in discovering that this was the required set of services. Discovering the required set was problematic because:

- There was no clear marker distinguishing score server traffic from attacker traffic.
- If the score server was dissatisfied at any point, it abruptly halted the poll sweep for that team. Thus we learned at most one bit of information about what the score server wanted on each pass.

We eventually despaired of ever getting the Immunix server to make the score server happy, and launched the reference system in desperation late Friday night. This turns out to have been the key to our success:

- The attackers on other teams were sufficiently accustomed to our server being difficult to pene-

trate that they did not immediately notice that we had launched the vulnerable reference system.
- The score server was immediately satisfied with our services, allowing us to score a few points and raise our morale.
- Critically, observing a successful pass of the score server gave us the information we needed: the required services.

The services required by the score server were:

1. The score server adds a user to your server via the `adduser` CGI script.
2. The score server fingers that user.
3. The score server logs in via FTP and deposits a file on your server.
4. The score server reads the deposited file from your server via HTTP.
5. The score server sends that user some e-mail via SMTP.
6. The score server POP's mail from that user.
7. The score server logs in to your server via telnet as the created user, cats the file deposited in step 3, deletes the file deposited in step three, and then pauses.
8. The score server logs in via FTP and uploads a PERL script.
9. The paused telnet session resumes, invoking the PERL script uploaded in step 8. This script computes the MD5 of flag on the server to award the score.
10. The score server deletes that user via the `deleteuser` CGI script.

Thus it was not until the beginning of Saturday morning that the Immunix server satisfied the score server. We were in 6th place of 8, with a lot of catching up to do.

4.2 Saturday: Immunix Works

A working Immunix server made a large difference. We had a secure, unassailable base to work from. Our attack team had been successfully penetrating other teams throughout Friday's play, but because of the rule that the home server had to satisfy the score server to earn an "own3d" point, they had been unable to score on Friday, and could only deny other teams their "home" points. Once the Immunix server was up, not only did we have a near guaranteed stream of "home" points, but we also enabled our attack team to score substantial additional points. We had reasonable offense and near-perfect defense.

Having a highly secure server changed the way we defended our system. While other teams employed the "massive human intrusion" defense of killing off hostile intrusions (sometimes measured in intrusions per second) we were able to contemplate issues as they came up. We allowed attackers to log into our server, but granted them only shells confined with the SubDomain mechanism so they could not do very much. At some points we even taunted the intruders by writing to their tty. Issues could be dealt with in a considered manner.

Throughout the game, there were also "distraction games" to be dealt with. These were various contests aside from the CtF game itself, intended to draw resources away from the teams. The distractions were intended to model the real-world distractions that defenders must deal with, while retaining a security theme. The prize for winning a distraction game was to get one technical question fully answered by the Ghettohackers, so playing the distraction games was worth while. Distraction games included:

Dumpster Diving/Steganography: The Ghettohackers asked one representative from each team out into the hallway, and then announced that there were two prize boxes in a dumpster on the other side of the hotel, and a foot race ensued. We recovered one of the boxes[1], which turned out to be a floppy disk filled with soft-core pornography images and family photos, with other data steganographically encoded into the photos. While we discovered that one of the images also appeared to be a DOS boot sector, we never recovered any useful data from the floppy.

Defcon Shoots: On Saturday morning, a contingent of firearms enthusiasts went out into the desert to destroy an assortment of obsolete hard drives with various .30 to .50 calibre weapons. The hard drives were color-coded to match the CtF teams, and the team whose drive was most thoroughly wrecked won that distraction game. We did not participate.

Lock Picking: Teams were handed small boxes with a door lock on it, and told to recover the contents without damaging the box. A race to recruit assorted lock-picking experts (who were on site to give a talk on lock picking) ensued. We recruited one of the better lock pickers, and recovered the contents (a color-coded pen). Unfortunately, we were *second* to recover

1. Resulting in a brief visit to an emergency ward for Crispin Cowan for respiratory distress. Running in Nevada in August when you are not used to it is not recommended.

Table 1: Final CtF Score

Team	Score	Points	Penalties
Orange (Digital Revelation)	54.3764	64	9.6236
White (Immunix)	**51.1160**	**55**	**3.8840**
Brown	48.2203	90	42.7797
Green (oxooffoo)	40.1943	46	5.8057
Yellow (ChaosComputerCoderZ)	22.1865	43	20.8135
Red (CRMP/Naval Postgraduate School)	6.1215	17	10.8785
Blue (unix-monkey.org)	-22.7039	30	52.7039
Purple (The Network Idiots)	-24.5107	5	29.5107

the contents, and so did not get the prize of a technical question answered. We *did* get a 2nd prize, which was that our bandwidth penalty was waived if we used a particular source port. However, our attackers were sufficiently skillful and subtle that this perk was not useful to us.

BSA (Business Software Alliance) Audit: Mid-day on Saturday, the Ghettohackers donned "BSA" t-shirts and conducted a surprise audit for "pirated" software. The main impact of this "inspection" was that all players were required to step away from their keyboards while the inspectors did their work. This cost us one score point, as we were in the middle of fixing a configuration issue when the BSA audit hit, and the score server polled our (temporarily broken) server while we were away from the keyboard.

These issues aside, the security of our Immunix defense and the strength of our attack team caused our score to advance rapidly from 7th to 1st place.

4.3 Sunday: Enter Webmin and PERL Fork Bombs

Late Saturday night, the Ghettohackers announced that a change was being implemented, and now the score server was going to be imposing new requirements. That new requirement was Webmin [3] the open source web management interface.[1]

Webmin presented a challenge to the defenders in general because it is famously vulnerable. Webmin presented a challenge to Immunix in particular, because as a far-reaching management interface, it requires permission to do a great many dangerous operations. Confining such a complex system with SubDomain is problematic, and not feasible in a very short period of time.

We spent some time overnight to produce a nominal SubDomain configuration for Webmin, but we failed. Our Webmin configuration was good enough to make the score server happy, but *post hoc* analysis revealed that all the score server wanted from Webmin was the ability log in to the Webmin interface; the score server never actually manipulated anything with Webmin. Conversely, we failed to protect the system from Webmin's ability to reboot the server, allowing attackers to re-boot our server at will.

A second problem emerged on Sunday: PERL fork bombs. We had effectively confined the CGI scripts such that they could run the PERL interpreter, but could not do damage to the server. However, PERL has the capability to fork itself, allowing the attacker to upload a PERL script that forks endlessly, consuming all of system memory and swap space, rendering the system inoperative. Worse, the only way to regain control of our server is to cut power, forcing a `fsck` upon reboot, resulting in substantial downtime.

These problems caused us to lose ground to our competitors in the 4 hours of play on Sunday. In the final analysis shown in Table 1, we placed second, 3.5 points out of 54 behind the Orange team.

However, we emphasize that at no time did attackers compromise the Immunix server: they could log in, but could not access files we did not want them to access.

1. Which, incidentally, competes with one of WireX's commercial products.

Table 2 shows the blow by blow "own3d" scores throughout the game. Of the 178 total score rounds in the game, only the 13 rounds shown in Table 2 were recorded with a flag other than the defender's on the machine. The entries marked "own3d" are cases where the attacker was awarded a point. Repeated entries indicate repeated successful attacks. The cases marked "almost own3d" are where the attacker's flag was recorded as being on the victim's server, but that the victim's server was not fully satisfying the score server, and so no point was awarded.

We have no data for cases where the victim's server was own3d, but no point was awarded because the *attacker's* server was not satisfying the score server. We believe there are many cases of the latter due to Immunix being down all of Friday. The one instance where white (Immunix) was own3d was during the time on Sunday when we were running the reference server while waiting for the Immunix server to `fsck`.

5 Strategic Analysis

Here we analyze how our strategy succeeded Section 5.1, and failed in Section 5.2.

Table 2: Blow by Blow "own3d" Results

Round	Result
89	blue almost own3d red
91	white own3d brown
99	white almost own3d red
103	white own3d red
104	white own3d red
105	white own3d red
109	brown own3d red
130	green own3d blue
134	green almost own3d blue
134	orange own3d red
136	orange own3d red
137	green almost own3d blue
157	green own3d white

5.1 Success

Our primary goal in playing the Defcon CtF game was to validate the survivability properties of the Immunix system, comprised of technology components developed under various DARPA programs over the last five years. We made the strategic decision to never place an Immunix server on the play network that was not robust against attack. This strategy succeeded, in that the Immunix server was exposed to substantial attacks, but was never compromised.

A secondary, unanticipated success was demonstrating the relative ease with which fairly complex and unknown software can be ported to Immunix and contained by SubDomain. While it took 12 hours to port and profile the reference system's applications to Immunix, and those hours were frustrating and expensive in terms of CtF points, it is also an accomplishment that it *only* took 12 hours to reproduce an unknown set of services with no specification to work from and confounded feedback on whether the services are correct.

5.2 Failure

Our strategy failed, in that we could have made some decisions differently, without compromising our primary objective (validate Immunix):

- We lost many "home" points during the delay of the first day when we were not operational.
- We lost more "own3d" points, because we successfully penetrated other servers, planted our flag, but did not score a point because our server was not operational.
- It took us an excessive amount of time to get our server fully interoperating with the score server, because the score server would abort its run on the first failure.

What we *should* have done was launch the reference system immediately (like everyone else did) and sniffed the network to more quickly learn what the score server wanted. This is in fact what the Ghettohackers recommended that we do, but we resisted for several hours for fear that the Immunix server would *never* get to play, invalidating the experiment. It is only in hindsight that we realized that launching the reference server *accelerated* the Immunix server into play. This is a direct result of the lack of any specification for the required score server functions, requiring the use of a working example to successfully determine what the score server needed.

6 Tactical Analysis

Here we analyze the tactical play: attacks that we deployed in Section 6.1, and attacks that were deployed against us in Section 6.2.

6.1 Attacks We Deployed

We successfully deployed buffer overflow attacks against `ftpd` (early on) and `sendmail`, race attacks against `at`, and configuration attacks against Webmin (toward the end). Once we had penetrated a system, we had a broad arsenal of malicious code and actions to deploy:

- Change the shell for the root account to `/bin/halt` preventing the victim from logging in to their own server.
- Remove "`kill`", "`killall`", "`shutdown`", "`reboot`", etc., frustrating the human intrusion detection & response approach.
- Create administrative accounts with names like 'bin, 'adm', and ' bind' , which the human intrusion detectors tended not to notice.
- Trojaned `ps` to not see our login names.
- Make `setuid root` copies of `sh` and hide them in unusual places in the file system, such as `/dev`.
- Killed `syslogd`.
- Replaced their flag, with a script set to repeatedly copy our flag to the reference location with an official-sounding name such as `/sbin/pklogd`.
- Spam' bot: a particularly creative attack program. Because there is a penalty attached to bandwidth, we would infect victim A with a spam'bot that would send e-mail to all of the other teams. Because the bandwidth cost was per connection, the spam'bot would send 1-byte e-mails. Because the spam'bot imposed very little load and did not disrupt either the flag or the required services, teams would often not notice it was there. There was also a spam' bot variant that implemented a similar attack using telnet instead of e-mail.

6.2 Attacks Deployed Against Us

Attackers often got login shells on our server, but these shells were ineffective. We had changed the adduser CGI to create accounts with a default shell of /bin/fubush which was in turn a link to /bin/bash, but with a tight profile around /bin/fubush so that it could *only* execute the commands needed by the score server. Login shells were so harmless to us that we are considering printing our root password on a large sign and hanging it from our team table at next year's game.

An attacker did successfully hack our `telnetd`. There was a public vulnerability in telnetd announced in summer 2001 [17]. WireX did not bother to patch this vulnerability, because we did not anticipate anyone seeking a secure operating system installing `telnetd`. However, the score server required telnet, so we installed `telnetd`, and provided it with a SubDomain profile. The result was that the attacker was able to compromise our telnet service, but not get any farther than that, prevented by the SubDomain profile. Patching our `telnetd` restored our telnet service.

As described in Section 4.3, attackers eventually discovered that they could fork-bomb PERL scripts on our server, resulting in a denial-of-service.

We suffered some self-inflicted attacks. In the zeal to ward off attackers, defenders would sometimes kill login shells started by the score server, resulting in the loss of a point for that round.

We observed numerous ill-advised attacks, such as repeated `nessus` [11] scans, and bulk exploit scripts that included Microsoft IIS exploits. In light of the CtF game's bandwidth penalty, this kind of shotgun attack is ill-advised, as it does more damage to the attacker than the victim.

7 Future Work

Here we present our philosophical conclusions from this experience. Section 7.1 describes potential improvements for Immunix. Section 7.2 describes how to improve our game-playing strategy. Section 7.3 suggests how the Ghettohackers (or others) might improve upon this game.

7.1 Improving Immunix

While it is impossible, in principle, to completely defend against denial-of-service attacks, the successful DoS attacks against Immunix suggest that more should be done:

Continue *Guard Development: Unlike access control schemes like SubDomain, the *Guard tools (StackGuard, FormatGuard, and RaceGuard) halt exploits in their tracks. This substantially reduces the potential DoS that an attacker can deploy against a vulnerable service.

Resource Management: The PERL fork bomb showed that some kind of resource management should be built into SubDomain. We are still working on what

the semantics of this resource management should be.

7.2 Improving Our Gaming Strategy

To improve our game playing strategy:

Launch the Reference System Immediately: So long as the CtF game is characterized by having to guess the difference between "good" (score server) and "bad" (attacker) traffic, there is a lot of value in launching the provided reference system early to learn from its interactions with the score server, even if this does cost some "own3d" points.

VMWare Proficiency: The CtF game is likely to continue using VMWare to distribute the reference systems, so greater proficiency with VMWare is called for. We lost some points due to simple delay in configuring VMWare networking correctly.

Replicas: Replication is a problematic approach to system survivability, because it is very difficult to prevent the attacker from deploying the same attack across the replicated systems [7]. However, replicas do appear to be useful in the CtF context, because of the simultaneous presence of these factors:

Low-latency scoring: time is of the essence to recover the team's server if it crashes, or you will miss a score round.

Massive human response: When the attacker cracks the first replica, the humans can jump into action to ensure that the same attack is not deployed against other replicas. This is only feasible with substantial manpower watching the servers being attacked. It is not amenable to automated response, because it is impossible to anticipate what the attackers will have done. If attacker action could be effectively anticipated, then it could be prevented outright.

Better Logging: As a product design strategy, Immunix has focussed on intrusion prevention, and has relatively little in the way of event logging. But in the CtF setting, where you have ample manpower available but not very much time, better forensic tools would help to recover quickly.

Bring More Hardware: We *thought* we had brought enough hardware, in that we brought 100% redundant systems (duplicate laptop servers and duplicate hubs). In practice, we needed multiple servers just to play. We will bring at least three playing servers next time.

7.3 Improving CtF

This year's CtF game was outstanding, so these suggestions are not to be taken as criticisms. However, they might result in an interesting *different* game, if not necessarily a better game:

Use cryptographic protocols:, If the score server is authenticated using cryptographic means (so that attackers cannot replay the authentication) then the teams can be required to support much more sophisticated (and even stateful) services for the score server. With plain text authentication as in this year's game, the players depend on the predictable behavior of the score server to guess the difference between score server traffic and attacks. If score server behavior remains static, then it becomes feasible to build a satisfactory playing server using an expect script, and not actually have to provide any services at all.

8 Conclusions

Prior to the game, our expectation was that keeping Immunix from being compromised would be difficult, and we had no expectation to place high in the score. We are delighted at keeping attackers from compromising our server, and surprised to have come within a nose of winning the game. This experience has raised our confidence in the security of our platform, as well as showing us where it needs to be improved.

9 Acknowledgments

The Immunix team comprised 4 WireX staff (Seth Arnold, Steve Beattie, Crispin Cowan, and Chris Wright) with a great deal of help from John Viega. An additional 20 volunteers joined the Immunix team at the conference. Those who have agreed to be listed here are Jay Beale, Pravir Chandra, "cubes", Bob Fleck, "hey_man", Cliff Jolly, Toby Kohlenberg, "minion", Adam Shand, Ed Skoudis, Paolo Soto, and Kathy Wang. Chandra and Fleck in particular did excellent work attacking.

References

[1] M.Bishop and M. Digler. Checking for Race Conditions in File Accesses. *Computing Systems*, 9(2):131–152, Spring 1996. Also available at url http://olympus.cs.ucdavis.edu/ bishop/scriv/index. html.

[2] Kalou/Pascal Bouchareine. Format String

Vulnerability. url http://plan9.hert.org/papers/format.html, July 18 2000.

[3] Jamie Cameron. Webmin. url http://www.webmin.com/, September 2002.

[4] Crispin Cowan, Matt Barringer, Steve Beattie, Greg Kroah-Hartman, Mike Frantzen, and Jamie Lokier. FormatGuard: Automatic Protection From printf Format String Vulnerabilities. In *USENIX Security Symposium*, Washington, DC, August 2001.

[5] Crispin Cowan, Steve Beattie, Calton Pu, Perry Wagle, and Virgil Gligor. SubDomain: Parsimonious Server Security. In *USENIX 14th Systems Administration Conference (LISA)*, New Orleans, LA, December 2000.

[6] Crispin Cowan, Steve Beattie, Chris Wright, and Greg Kroah-Hartman. RaceGuard: Kernel Protection From Temporary File Race Vulnerabilities. In *USENIX Security Symposium*, Washington, DC, August 2001.

[7] Crispin Cowan, Heather Hinton, Calton Pu, and Jonathan Walpole. The Cracker Patch Choice: An Analysis of Post Hoc Security Techniques. In *Proceedings of the 19th National Information Systems Security Conference (NISSC 2000)*, Baltimore, MD, October 2000.

[8] Crispin Cowan, Calton Pu, Dave Maier, Heather Hinton, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, and Qian Zhang. StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks. In *7th USENIX Security Conference*, pages 63–77, San Antonio, TX, January 1998.

[9] Crispin Cowan, Perry Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole. Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade. In *DARPA Information Survivability Conference and Expo (DISCEX)*, January 2000. Also presented as an invited talk at SANS 2000, March 23-26, 2000, Orlando, FL, url http://schafercorp-ballston.com/discex.

[10] "Solar Designer". Non-Executable User Stack. url http://www.openwall.com/linux/.

[11] Renaud Deraison et al. Nessus. url http://www.nessus.org/, August 2002.

[12] Fyodor. Nmap: Network Mapper. url http://www.insecure.org/nmap/, August 2002.

[13] "Ghettohackers". The Ghettohackers. url http://ghettohackers.net/.

[14] Robert Lemos. Putting Fun Back Into Hacking. CNet news.com, url http://news.com.com/2100-1001-948404.html, August 5 2002.

[15] Tim Newsham. Format String Attacks. Bugtraq mailing list, url http://www.securityfocus.com/archive/1/81565, September 9 2000.

[16] "Aleph One". Smashing The Stack For Fun And Profit. *Phrack*, 7(49), November 1996.

[17] Scut. Multiple Vendor Telnet Daemon Vulnerability. url http://online.securityfocus.com/archive/1/197804, July 18 2001. Bugtraq.

[18] "tf8". Wu-Ftpd Remote Format String Stack Overwrite Vulnerability. url http://www.securityfocus.com/bid/1387, June 22 2000.