Black hat knowledge for white hat programmers [closed]

Asked 15 years, 8 months ago Modified 5 years, 9 months ago

Viewed 5k times

72

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 11 years ago.

Improve this question

There's always skepticism from non-programmers when honest developers learn the techniques of black hat hackers. Obviously though, we need to learn many of their tricks so we can keep our own security up to par.

To what extent do you think an honest programmer needs to know the methods of malicious programmers?

security

Share

Improve this question

Follow

edited Mar 22, 2017 at 16:39

user149341

asked Apr 21, 2009 at 13:29

Dinah

54k ● 30 ● 134 ● 149

2 I think the better question would be "How do I start acquiring the knowledge that the black hats have already gotten?" That is, is there some starting point for their reverse engineering tricks etc. that an honest developer might start with? – Onorio Catenacci Apr 23, 2009 at 19:44

2 I'd say "to the full extent". How else can you prevent attacks unless you know exactly how they are performed? – mpen Dec 18, 2009 at 8:08

21 Answers

Sorted by: Highest score (default)

45

At the end of the day nothing the 'black hats' know is criminal knowledge, it's just how the knowledge is applied. Having a deep understanding of any technology is valuable as a programmer, it's how we get the best out of the system. It's possible to get by these days without knowing the depths as we've more and more frameworks, libraries and components that have been written using such knowledge to save you having to know everything but it's still good to dig from time to time.

Share Improve this answer

Follow

answered Apr 21, 2009 at 13:39

Lazarus

39

I'm coming in late on this, as I just heard about it on the podcast. However, I'll offer my opinion as someone who has worked on the security team of a software company.

We actually took developer education very seriously, and we'd give as many teams of developers as possible basic training in secure development. Thinking about security really does require a shift in thinking from normal development, so we'd try to get developers thinking in a how-to-break-things frame of mind. One prop we used was one of those home safes with the digital keypad. We'd let developers examine it inside and out to try to come up with a way of breaking in to it. (The solution was to put pressure on the handle while giving the safe a sharp bash on the top, which would cause the bolt to bounce on its spring in the solenoid.) While we wouldn't give them specific black-hat techniques, we'd talk about the implementation errors that cause those vulnerabilities -- especially things they might not have encountered before, like integer overflows or compilers optimising out function calls (like memset to clear passwords). We published a monthly security newsletter internally, which invited developers to spot security-related bugs in small code samples, which certainly showed how much they would miss.

We also tried to follow Microsoft's Security Development Lifecycle, which would involve getting developers to talk

about the architecture of their products and figure out the assets and possible ways to attack those assets.

As for the security team, who were mostly former developers, understanding the black-hat techniques was very important to us. One of the things we were responsible for was receiving security alerts from third parties, and knowing how difficult it would be for a black hat to exploit some weakness was an important part of the triage and investigation processes. And yes, on occasion that has involved me stepping through a debugger to calculate memory offsets of vulnerable routines and patching binary executables.

The real problem, though, is that a lot of this was beyond developers' abilities. Any reasonably sized company is going to have many developers who are good enough at writing code, but just do not have the security mindset. So my answer to your question is this: expecting all developers to have black-hat knowledge would be an unwelcome and detrimental burden, but *somebody* in your company should have that knowledge, whether it be a security audit and response team, or just senior developers.

answered May 4, 2009 at 1:32

Isvara

3,463 ● 2 ● 30 ● 46

1 That is an EXCELLENT answer. You're absolutely right, particularly in your last paragraph. – David Dec 3, 2010 at 13:34

39

I'm going to be a bit heretical and go out on a limb and say:

- You really need to talk to the sysadmin/network folks that secure their machines. These folks deal with the concept of break-ins every day, and are always on the lookout for potential exploits to be used against them. For the most part, ignore the "motivation" aspect of how attackers think, as the days of "hacking for notoriety" are long gone. Focus instead on *methodology*. A competent admin will be able to demonstrate this easily.

When you write a program, you are presenting what is (hopefully) a seamless, smooth interface to ${whatever-else-accepts-your-programs-I/O}. In this case, it may be an end-user, or it may be another process on another machine, but it doesn't matter. **ALWAYS assume that the "client" of your application is potentially hostile, regardless if it's a machine or a person.**

Don't believe me? Try writing a small app that takes sales orders from salespeople, then have a company rule that you need to enforce through that app, but the salespeople are constantly trying to get around so they can make more money. Just this little exercise alone will

demonstrate how a motivated attacker - in this case, *the intended end-user* - will be actively searching for ways to either exploit flaws in logic, or to game the system by other means. *And these are trusted end-users!*

Multiplayer online games are constantly in a war against cheaters because the server software typically trusts the client; and in all cases, the client can *and will* be hacked, resulting in players gaming the system. Think about this - here we have people who are simply enjoying themselves, and they will use extreme measures to gain the upper hand in an activity that doesn't involve making money.

Just imagine the motivation of a professional bot herder who makes their money for a living this way...writing malware so they can use other people's machines as revenue generators, selling out their botnets to the highest bidder for massive spam floods...yes, this really does happen.

Regardless of motivation, the point remains, your program can, and at some point will, be under attack. It's not enough to protect against buffer overflows, stack smashing, stack execution (code-as-data is loaded into the stack, then a return is done to unload the stack, leading to execution of the code), data execution, cross-site scripting, privilege escalation, race conditions, or other "programmatic" attacks, although it does help. In addition to your "standard" programmatic defenses, you'll also need to think in terms of trust, verification, identity,

and credentials - in other words, dealing with whatever is providing your program input and whatever is consuming your program's output. For example, how does one defend against DNS poisoning from a programmatic perspective? And sometimes, you can't avoid things in code - getting your end-users to not turn over their passwords to coworkers is an example.

Incorporate those concepts into a *methodology* for security, rather than a "technology". Security is a process, not a product. When you start thinking about what's "on the other side" of your program, and the *methods* you can employ to mitigate those issues, it will become much clearer as to what can go right, and what can go *horribly* wrong.

Share Improve this answer

Follow

edited Jun 15, 2010 at 23:32

answered Apr 21, 2009 at 14:01

Avery Payne

1,748 ● 2 ● 17 ● 31

19

To a large extent. You need to think like a criminal, or you're not paranoid enough.

Share Improve this answer

Follow

answered Apr 21, 2009 at 13:31

ScottStonehouse

24.9k ● 7 ● 33 ● 34

2 you can't think like that if you don't know how think like that. It's like "think about a LINQ solution without learning LINQ". It won't work. – dr. evil Apr 21, 2009 at 13:41

17 Thinking like a criminal involves acquiring knowledge. Criminals go through a learning process too. – Ben S Apr 21, 2009 at 14:09

11 And I thought it just involved strangling kittens.... – Quibblesome Apr 21, 2009 at 15:08

5 (-1) Although your comment is a step in the right direction, you haven't actually said anything that would be useful to asker. I.E. how is he to 'think like a criminal', how is he to 'learn everything they do'. – DevinB Apr 21, 2009 at 17:21

1 @Scott, I have no problem with 'thinking like a criminal' as a solution to this issue. I simply think that your answer does nothing to help the person who asked the question. – DevinB Apr 28, 2009 at 13:28

18

> To what extent do you think an honest programmer needs to know the methods of malicious programmers?

You need to know *more* than them.

Share Improve this answer

Follow

answered Apr 21, 2009 at 13:38

Bill the Lizard

405k ● 211 ● 572 ● 889

read my answer, you can simply see most of the time it's just not possible unless you got too much time in your hands to pour in it. – dr. evil Apr 21, 2009 at 13:42

3 I disagree. If you're a web programmer you need to know all the ways your programs are going to be attacked in order to protect them. You can (and should) hire professionals to help you, but it should be a part of your job to learn the security techniques used in whatever type of programs you're paid to develop. – Bill the Lizard Apr 21, 2009 at 14:28

13 There is big difference between cracking into something and protecting against it. If you write parameterised SQL Queries, it's done, code secure against SQL Injeciton. Therefore you don't need to know 10 different ways of how to exploit SQL Injections. If you do quick search on exploiting SQL Injections you can simply there 4-5 different main ways based on the context and so many other possibilities based on backend database.But as a developer knowing 10 different ways is not going to help you. – dr. evil Apr 21, 2009 at 14:37

5 I see your point, and I agree somewhat. You don't need to know how to execute 10 different kinds of attack, but you do need to know how to protect against all of them. It's not always as easy as one solution, parameterized SQL queries, that covers all attacks. – Bill the Lizard Apr 21, 2009 at 15:02

13

I do work as a security guy not a developer and based on my experience I can simply say you can't learn stuff as good as black hat or professional white hats do unless it's your second profession. It just too time consuming.

Most important bit though seeing some bad guys or professionals on action and understanding what are the

possibilities and the impact of unsecure code.

So by learning some tricks but lots of them one might get feeling of "false sense of security" because he or she can't hack. Although a better skilled attacker might hack the same thing within minutes.

Having said that, as soon as you keep this in mind I think it's good to learn some attacks, fun and quite educational to learn how to break stuff.

Share Improve this answer

Follow

edited Apr 21, 2009 at 13:49

answered Apr 21, 2009 at 13:35

dr. evil

27.2k ● 37 ● 134 ● 202

2 I don't think learning a few techniques will leave you feeling more secure than learning nothing at all. If you learn how these things are done, you should be scared, not lulled into a false sense of security. – ScottStonehouse Apr 21, 2009 at 13:40

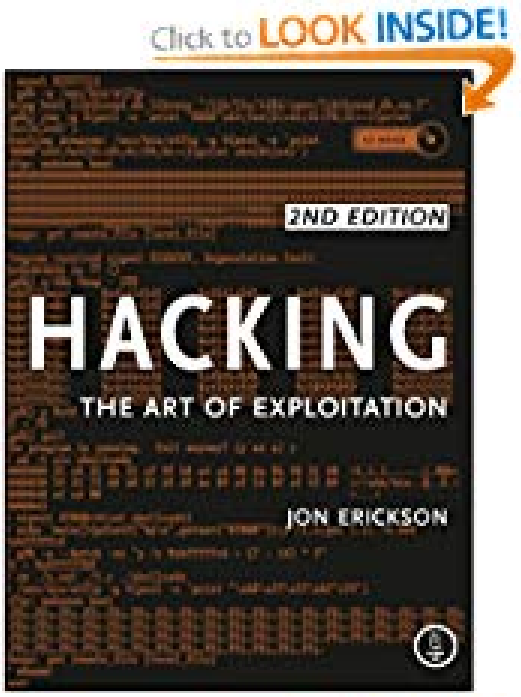
1 Good point though, that you can't really know security unless it's your full time job. It's not my full time job, and that just makes me more paranoid. – ScottStonehouse Apr 21, 2009 at 13:44

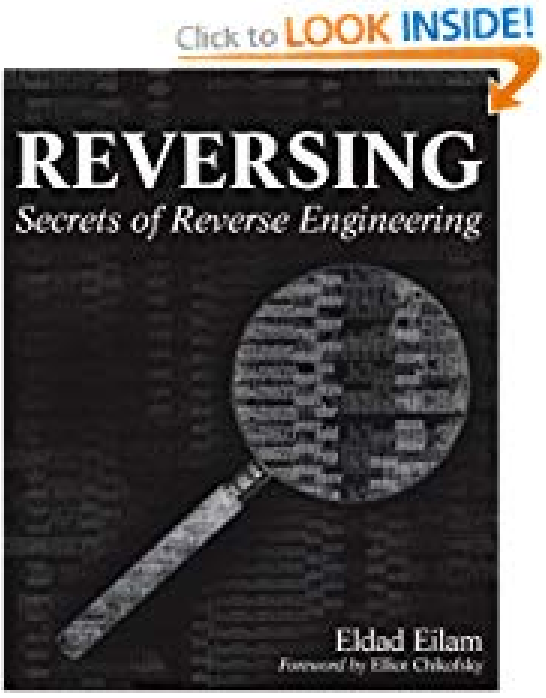
1 You right, as soon as you aware of the fact that you don't know everything about that subject, it's always good to get your hands a little bit dirty :) – dr. evil Apr 21, 2009 at 13:48

Just updated the answer to clarify / emphasize the last sentence. – dr. evil Apr 21, 2009 at 13:49

10

Definitely learn the dark side. Even if you don't learn the actual techniques, at least make the effort to learn what's possible.





Good resources to learn the tricks of the trade are Reversing: Secrets of Reverse Engineering and Hacking: The Art of Exploitation. They're written for both sides - these could be used to LEARN how to hack, but they also give ways to prevent these kinds of attacks.

Share Improve this answer

Follow

edited Mar 12, 2019 at 11:00

Glorfindel

22.6k ● 13 ● 89 ● 116

answered Apr 22, 2009 at 13:24

ryeguy

66.8k ● 60 ● 201 ● 263

9

It pays to be as "innocent as doves, and as wise as serpents," and learn techniques that folks with nefarious purposes do. That said, such knowledge should be used carefully. "With great power comes great responsibility".

Share Improve this answer

Follow

answered Apr 21, 2009 at 13:37

J. Polfer

12.5k ● 10 ● 56 ● 83

8 Quoting the Bible and Spider-Man together. I like it. – Jamison Dance Apr 21, 2009 at 14:52

9 With great power comes great electricity too! – Joe Phillips Apr 23, 2009 at 14:28

8

One word of caution: State of Oregon vs. Randal Schwartz.

Having been a small part of investigating two separate incidents at our site, I'd say the odds of learning about an exploit before it's used against you are vanishingly small. Perhaps if you dedicate your career to being a white hat you'll stay on top of all the potential holes in most of the popular hardware/software stacks. But for an ordinary programmer, you are more likely to be in reaction mode.

You do have a responsibility to know how your own software could be hacked and a responsibility to stay reasonably up-to-date with third-party software. It would be good to have an emergency plan in place to deal with an attack, especially if you are a high-profile or high-value target. Some places will want to shut a hole immediately, but our site tends to leave certain holes open to assist law enforcement in catching the perpetrators. The IT security team occasionally announces internally that it will be conducting a port scan so that SA's don't freak out about it.

Share Improve this answer

Follow

answered Apr 29, 2009 at 23:37

Jon Ericson

21.5k ● 12 ● 102 ● 151

> So it sounds like knowing how to do it is probably okay, but actually doing it without permission is absolutely not okay!
> – PearsonArtPhoto Dec 14, 2015 at 19:53

5

One skill that is often missed is social engineering.

Many people simply do not recognize when they are being conned. At a prior company a VP ran a test by having three (female) temp workers in a conference room call programmers and sysadmins and work from script to try and get someone to grant access or reveal passwords. Each of the temps got access to something in the first hour of calls.

I bet if a similar test were run at any mid to large sized company, they would get the same results.

Share Improve this answer Follow

answered Apr 23, 2009 at 19:29

sal

23.6k ● 15 ● 69 ● 85

5

I think part of 'coding defensively' includes knowing malicious techniques, but at the same time, you don't necessarily need to know all of the techniques in order to defend against them effectively. For instance, knowing about buffer-overflow attacks isn't the reason to try and protect your buffers from overflowing. You protect them from overflowing because if they do, it could wreak havoc in your program regardless of whether it's a bug or an attack.

If you write very thoroughly checked and well architected code, then the malicious attacks will be unable to penetrate, because good architecture should automatically lock out side-effects and unauthorized access.

However, that last paragraph assumes that we have a perfect job where we are given incredible amounts of time to make our code *just right*. Since such a job doesn't exist, then knowing the malicious techniques is a good shortcut, because it means that although your code isn't perfect, you can create 'un-workarounds' for those exploits to make sure that they do not get through. But,

those don't make the code better, and they don't make the application better.

Ultimately, knowing malicious exploits is something that is good to be aware of, but 95% of them will be covered by simply making sure you adhere to best practices.

Share Improve this answer

Follow

edited Dec 18, 2009 at 7:58

Dinah

54k ● 30 ● 134 ● 149

answered Apr 21, 2009 at 13:40

DevinB

8,307 ● 9 ● 46 ● 55

5

Design for evil. "When good is dumb, evil will always triumph."

In short, if you don't think like a criminal, that doesn't mean the criminals won't.

Share Improve this answer

Follow

edited Jun 15, 2010 at 16:56

Kevin Panko

8,504 ● 19 ● 51 ● 63

answered Apr 21, 2009 at 13:34

Brian

25.8k ● 18 ● 86 ● 178

I personally don't see the technical difference. Sure the motives are different but the technical game is the same.

4

It's like asking what kind of warfare the "goodies" need to know about.

The answer is all of it, even if they don't actively practice it.

Share Improve this answer

Follow

answered Apr 21, 2009 at 13:34

Quibblesome

25.4k ● 10 ● 62 ● 104

3

One of the techniques the White Hats need to learn is how to test/mitigate/think in terms of social engineering, because the biggest security threat is people.

White Hats are good at manipulating bits, but people are the ones manipulated far more often by the Black Hats.

Share Improve this answer

Follow

answered Apr 21, 2009 at 14:44

Gavin Miller

43.7k ● 22 ● 126 ● 191

1

I'm going to take the controversial stance and say that there's some black-hat knowledge that you don't need to be a good white-hat hacker. A doctor doesn't need to know how to genetically engineer a virus in order to effectively treat illness.

Share Improve this answer

Follow

answered Apr 22, 2009 at 22:20

Gabriel

19 ● 2

Software development isn't (hopefully) like a doctor treating illness. Your doctor might, in fact, only sign off on nurse to administer a flu shot you requested. Furthermore, in general doctors don't have an effective treatment for viruses. However, some of the people involved in the development of the vaccine itself most likely do posses the knowledge to engineer a virus. – Marsh Ray Oct 7, 2009 at 21:44

1

we white hats and gray hats need to be good at a million things those black hats and skiddies only have to succeed with one thing

Share Improve this answer

Follow

edited Oct 7, 2009 at 21:28

Dinah

54k ● 30 ● 134 ● 149

answered Apr 21, 2009 at 14:14

Jan-Willem Hoekman

65 ● 1 ● 8

5 This has also been expressed as "We have to be lucky every time, they (hackers) only have to be lucky once" – Richard Ev Apr 22, 2009 at 13:19

true that is the same sentence only in a different way – Jan-Willem Hoekman Nov 29, 2010 at 11:16

1

Basically almost all security exploits used by hackers are bugs in the code introduced by poor programming style or disciplines. If you write code to protect against bad data

and invalid call operations you'll block the majority of security vulnerabilities in your code.

If you're interested to protect your code from hacking/abuse/etc. you'll spend way too much time on it. Just buy a package to protect the basics and just move on.

Share Improve this answer

Follow

edited Jan 7, 2010 at 18:39

Dinah

54k ● 30 ● 134 ● 149

answered Apr 29, 2009 at 21:14

Paul Alexander

32.3k ● 16 ● 99 ● 151

0

You have to understand the methods the 'bad guys' use, so some understanding is mandatory.

For the average developer I think it is enough to grok the basic priciple of what they are doing in order to avoid creating vulnerabilities in their projects.

For somebody who works in a security relevant area (banking comes to mind, or credit card data from an online shop), a deeper understanding is required. Those developers need to go 'under the hood' of how a 'bad guy' operates and which techniques he uses.

Share Improve this answer

Follow

answered Apr 21, 2009 at 13:40

Treb

20.3k ● 8 ● 59 ● 88

0

To the point where by learning their ways he starts to think in their direction. And then he must choose which side he wants "to belong to".

There is nothing malicious in technology itself ... knowledge is pure ... it's how you use it that determines how it shall be looked upon.

Share Improve this answer

Follow

answered Apr 21, 2009 at 13:51

Rook

62.4k ● 53 ● 168 ● 247

0

2 sides of the same coin. Other than intent -- what's the question? Same skills, different implementation.

Share Improve this answer

Follow

edited Nov 12, 2009 at 17:47

Dinah

54k ● 30 ● 134 ● 149

answered Apr 22, 2009 at 13:07

TheAntsAreMyFriends

4 Nah - stepping through a debugger to calculate memory offsets of vulnerable routines and patching binary executables is a different set of skills from checking your inputs to prevent buffer overflows (for example). A programmer should understand the model, but the black hat has to know how to implement the attack. – guns Apr 22, 2009 at 22:38

1 @guns: this is one of the best examples yet. Please put it in a real answer so it can be upvoted and maybe accepted.
– Dinah Apr 23, 2009 at 14:22

0

When I hear the word blackhat, I think of someone who uses knowledge of computers to break into banks and do other mischievous things. A whitehat knows *everything* that the blackhat knows, but just doesn't do anything bad with it.

Therefore, you don't have *care* or *know* what a "blackhat" is to be secure...

Knowing how a blackhat *thinks*, when you're already an equivalent whitehat doesn't help squat. It's like knowing, "John wants to break into my house and steal my iPod music". If you really cared about your iPod music, you should have had it secure anyways.

Share Improve this answer

Follow

edited Mar 12, 2010 at 2:23

Dinah

54k ● 30 ● 134 ● 149

answered Nov 23, 2009 at 7:14

Longpoke

12.6k ● 4 ● 52 ● 55