Blood in the Water (Transcript of Discussion)

Sandy Clark

University of Pennsylvania

A couple of years ago when we were analysing voting machines we came across a question for which we didn't have an answer, namely if these machines are so bad why aren't they being attacked left and right? These machines were full of vulnerabilities, they were trivial to exploit, and yet it strikes me now there's been no documented case of an attack on any voting system by exploiting a software or a hardware vulnerability.

Peter Ryan: Perhaps you're coming onto this anyway, but there have been documented attacks on eVoting systems.

Reply: Yes, we'll explore that in just a second. So there's some sort of honeymoon, there's some sort of protection that these machines are enjoying right now, and we wanted to find out what it was. We went into possible explanations including things like lack of incentive (do you go to any bother on a US election), or maybe the bad guys are just a lot better at this than we are (that's possible, but I have no way to test it).

It did occur to us that one major difference between eVoting computer systems and other kinds of computer systems is that eVoting systems are really only accessible to the general public about two days a year, so maybe the secret to protecting voting systems, that right now is giving them this honeymoon, is the fact that attackers simply aren't familiar enough with them to have developed a set of attacks, or to make those attacks undetectable.

So we thought about familiarity, and one of the problems with that is that it goes against the whole standard of software engineering philosophy. This picture is from Brooks "The Mythical Man-Month", and the standard software engineering model believes that when we release a software product it has all of these bugs, and you find the easy bugs really fast, and you patch those bugs, and then you find the slightly harder ones and you patch them, eventually you get to a point where it's too hard to find any more bugs and you consider the software secure.

So what we did to try and find out whether familiarity mattered, whether or not the Brooks model worked, is we scraped the bug lists, we got over 30,000 vulnerabilities, we correlated them with bug traffic, we even manually checked about 3,000 of them, to make sure it was correct. Then we picked the popular server apps, the user apps, with both the Open Source and Closed Source code, and we correlated the vulnerability with the release state of the product, and the release state of each specific individual version of the product, and then we looked to see whether or not it matched Brooks model, or whether or not there was some sort of honeymoon effect going on. And what we found was a model that looked much more like this. There does indeed seem to be this

B. Christianson and J. Malcolm (Eds.): Security Protocols 2010, LNCS 7061, pp. 18–24, 2014.

honeymoon period where a client seems to be protected, and then this first primal vulnerability, that genesis vulnerability is found, and then like blood in the water attracting sharks, a whole bunch more vulnerabilities appear until you reach this sort of levelling off effect, and that could be as a result of end of life of software, or a new version is released, or something like that.

So if there's a honeymoon effect for software, where else might we see it? Because the interesting thing about the idea of familiarity is that it is extrinsic to the quality of the product itself. After a bunch vulnerabilities are found, and people get used to it, or the learning curve is over, then whether or not you've written a program seems to matter, but if voting machines are an example, and this honeymoon effect really exists, then properties that don't have anything to do with the quality of the software are what matters. So we thought maybe there'd be certain protocols, maybe some in firmware or hardware, or security architectures, and we just chose the first three that came to mind. So we chose, one security protocol, one algorithm, and we looked at the VMware, and surprisingly all three of them had something similar to a random effect going on.

So the first thing that we looked at was the Needham-Schroeder protocol, and at some basic differences between protocols and software. First of all, nobody releases a protocol that they know to be buggy and then says, stick it behind a firewall. Secondly, there's really no new version, no-one says, buy my new security protocol 2.0, it's got video, so the life-cycle of a security protocol is going to be much, much smaller. And in the case of Needham and Schroeder, when it was released, in the late 70s, early 80s, it had this long, long honeymoon. And then all of a sudden in 1994 there was an attack. In 1995 there was another attack, and then there were four all in one year. It was as if the security community had to learn how to attack the security protocol. Now especially important is that the attacks didn't attack until after the paper about the BAN logic came out, so once there was information on how to analyse a security protocol and a tool available to use to analyse it.

Virgil Gligor: There was a 1981 attack that was the first one which pointed out the vulnerability of the third message effect.

Reply: Thank you. So the sharks attacking came after we developed a tool to do it. And now of course there are things like protocol fuzzers, and attacking protocols is pretty well understood.

George Danezis: The 90s, when all these attacks are appearing is a special time, right, because these protocols are deployed and used on public networks.

Reply: For the first time in any wide scale.

George Danezis: Yes. We had lots of protocols in the 80s, but if you are an academic you were not quite sure if you are going to really have a career out of analysing them, breaking them, building better ones, or if they're just there, and they're using some military network, or some intranet, and you will never basically see them.

Reply: And that brings up an interesting point, are the sharks attracted because it's suddenly become a good research interest, is now the time that everybody should be looking at something? You see that a lot in the hacker community,

because someone will find a really interesting hack against something, and now everybody wants to jump on that bandwagon and then get props for finding another hack, so everyone starts to look at it.

So after we looked at the protocols we decided to look at hash algorithms, and we looked at MD5, HAVAL and the SHA family. And the SHA family is particularly interesting because of course it was the NSA who popped up with the message that SHA-0 was flawed, and you should put this here fix in and use SHA-1, and that was the blood in the water, that caused everybody to start looking at the SHA family and just to see if they could figure out what the NSA knew and they didn't. And then there is kind of a feature creep because you start to do larger and larger keys, we've got SHA-256, SHA-512, so there is kind of a new version of a hash algorithm released. But we really do understand now how you attack with hash algorithms. And the other interesting thing about the SHA family was Crypto 2005. I was there, I watched it on streaming video, and there was a paper in the main session that discussed attacks against SHA-1 on a reduced set. And then at the end of the conference in the rump session there were three more papers about SHA attacks, it's as if everyone had focused on it all at the same time, including that one by Wang et al which, when she'd finished talking, she got a standing ovation. So as far as we're concerned, with these algorithms anyway, the honeymoon is over, we're going to have to develop something new. All we're doing right now is getting by until we can find something new.

And the last thing that we looked at were virtual machines, and virtual machines have been given as the answer to everything. The answer to users who want to try something kind of risky, or they need special privileges, stick them on a virtual machine. You want to test malware, do it on a virtual machine, anything you want to do, virtual machines will save us. Except last summer, BlackHat, when Kostya Kortchinsky gave a wonderful talk on how you use the SVGA buffer as a memory leak to break out of the virtual machine onto a host, and to go back the other way around. And then a week ago last Tuesday, Core Security Technologies came out with the press release that they had broken Microsofts Virtual PC, in particular that they had been able to exploit the safe execution handlers, the memory randomisation, inside the virtual machine. If the operating system was running as the actual host, these would not be exploitable, but inside the virtual machine they are. And what's interesting about all these attacks even though they take place on two completely different software, you know, virtual PC and VMware, is that they both exploit the same idea, the fact that in order for a virtual machine to run it has to have special privileges that it might not otherwise have, and if you mess around with special privileges you can break out of the system.

So I expect if the pattern follows as it has been that we're going to see a number of attacks against virtual machines very soon. I think this is a flaw.

Alex Shafarenko: If I may interrupt. It is true if the virtualisation software is flawed, then there will be vulnerabilities that any guest operating the system could exploit. But the advantage is that once you've fixed the virtual hardware, you fix it for all virtual machines, whereas whatever you do with individual

operating systems, it will only work with that operating system, you'll have to do similar checks and certification processes with every other operating system. So virtual machines do not eliminate concerns, they concentrate them in one place.

Reply: That's an interesting observation, however, they're also running on another system, and you've got unexpected interactions that you can't plan for.

Alex Shafarenko: Only due to errors in design, not anything else.

Reply: Couldn't you say that about everything?

Alex Shafarenko: No, because every other design is variable, this one is fixed, once it's done, it's done. If you manage to find the last bug, it's still possible, it's a very localised design, because a virtual machine has a very focused job, it tends to its hardware, it doesn't do anything else. And it only emulates one type of hardware, because all these virtual machines ride on Macs for instance, like yours, right, they presume a certain hardware structure, and they don't give you a lot of choice or freedom. A physical PC would have a hundred choices, a choice of one of a hundred graphical cards, one of a hundred disk drives, etc, but in a virtual machine it would be just one set.

Virgil Gligor: Have you looked at the size of the virtual machine drivers and hyper drivers, in practice, and the size of the PCP for the virtual machines, like for example Xen, which is a pretty good design, over 120,000 lines of code VMM, and probably a million in the route from A0. Now if you don't have the variability there I don't know where you'd find variability.

Alex Shafarenko: What kind of variability are you talking about?

Virgil Gligor: Well variability in terms of software, and in terms of bugs that could be exploited, as Sandy points out. Even before, what she said, NSA ran an analysis study of VMware and Xen, and they found numerous bugs.

Alex Shafarenko: Due to the size of the product you will find bugs in it. My point is slightly different. You can freeze a virtual machine without detrimental effect on performance, because it's not a real machine.

Anil Madhavapeddy: Sorry I have to disagree. The issue for all emulation is the notion of power virtualisation, which is to run as much on the bare metal as you can, so you're actually running on the hardware without any protection. I think this is the reason for a honeymoon period: our assumption is changing, we're going from wrapping everything in a nice container to actually running stuff in the bare metal as much as possible.

But this leads to a very wide interface with the hardware, so it's actually not possible these days to take a virtual machine and run VM on an Intel chip, and run the same VM on an AMD chip, because you're running so much in the micro code concerned, and the actual interface is very different. So I think we're going to get a lot of attacks because our security researcher looked at this a decade ago and found a nice contained system, and now you come back to it even five years on and all of a sudden there's a new hardware support for VT and AVSVM.

Bruce Christianson: And the membrane was gone.

Anil Madhavapeddy: Exactly. The membrane's gone, it's just different now, so it's almost a case of security researchers don't have the bandwidth to look at all these evolving systems.

Reply: Well it's also that we set our perceptions based on one thing, and then it takes attackers to change our perceptions.

Alex Shafarenko: It also may be due to the fact that virtualisation is not added with the support of a firewall, because the hardware was not intended for any virtual machines, so with a suitable change in that design you may find your that your problems have a limited scope.

Reply: And that comes to a version of feature creep, well it can do this, why don't we see if it can also do this, it's the standard evolutionary model that everything follows.

Jeff Yan: Actually I can probably offer a possible explanation to the effect of the honeymoon effect extracted from a social psychology perspective. Basically human beings follow trends, our researchers are also humans.

Reply: Yes, I think that is a huge point, that's kind of the follower effect which we'll get to in a second.

So it looks like when you're looking at a security problem, during the very beginning of any vulnerability's life-cycle, or any product's life-cycle, extrinsic properties matter a lot more than we've ever given credit for. I think we as security community have long known intuitively, that vulnerabilities have different properties than bugs do. And I'm looking right now at some ways to test this. There are some people at ATT that have built a bug detector that's about 80% accurate on the software that they tested it on, as long as they have access to the source code and access to all of the bug tracking they can predict with 83%, 85% probability, which function is going to have the next bug. We're going to see if we can test that looking only at security vulnerabilities, and see if the same pattern applies. I'm hoping that there is some difference, I would like to believe that security vulnerabilities are different. The second point is that extrinsic properties are much more important. And something that we don't really think about, the first vulnerabilities found in a system are not necessarily the whole proof, and this again goes against the software engineering model.

When you're looking at something like a vulnerability, the first vulnerability in a product, and it's a result of legacy code, that's not "low hanging fruit", that legacy code existed in the previous version and wasn't found. And by the way, our forthcoming paper has a lot of information on this, about 50% of the vulnerabilities found in software are the result of legacy code, and a good percentage of the primal vulnerabilities are also the result of legacy code. And then the other possibility is, as you just said, people do what other people do, we follow trends. So this honeymoon effect we found, it seems to apply all over. Security protocols, crypto algorithms, seem to follow exactly the same pattern. Virtual machines follow the same pattern as software. But what's surprising is that superficially this isn't software, a security protocol is not software, there's some fundamental differences, which basically means that security patterns are alike no matter what, but the actual implementation is different.

And lastly, we asked the question, why should we care about the honeymoon at all? Well in the paper we just submitted we found the result that over the last years of looking at vulnerabilities and the products affected by them, nobody is winning the arms race. The last ten years the rate of vulnerability discovery has been just linear, which surprised all of us, we thought we would see spikes like an ECG. And in the past ten years there have been massive changes in the attacker and defender cyber-space. On the defender side there have been 300 plus new security initiatives. There's type-safe programming languages, there's automated patching, there's new types of firewalls and honeypots, and a lot of money spent to try and invent systems. And on the attackers side there are all of these great attacking frameworks that are available, many just point and click. And the minute a vulnerability is exposed it's announced the next day, you know, every patch Tuesday is followed by exploit Wednesday. There's also a change in the motivation, no longer is it just kids in their basements, but now you've got highly trained professionals profit motivated, or patriotism motivated, or whatever. So the balance of power shouldn't be the same, but any static system, any stalemate, always favours the attacker, because defenders have to defend against everything, but the attacker only has to find one new thing.

So we've got to figure out a way to change the arms race, and we only have two models. In the Centre for Disease Control model, you inoculate yourself against every known possible attacker that's out there and you stick yourself out in the wild and you wait to get attacked, and you clean yourself off and you do it again. The other model that we have (that tends to be used by military and our government) is the castle and moat model, where you build this huge fortress and you put a firewall outside and around you, and you stick yourself out there and you wait for the attack, and then you clean yourself off and you do it again. If we're going to change this, the only way to change the system is to change the rules of how we play, and that means we have to focus somewhere else. So let's focus at the beginning. Let's do offensive security codes. That's it. Any questions.

Dieter Gollmann: The first attack against the hash function was about 1993 by Hans Dobbertin on MD4, and at that time he told everyone who wanted to know that the same methods would work for MD5, but it would be difficult, and therefore for a very long time cryptographers preferred to do the easy things and not the difficult things, until they finally ran out of other problems to look at. So there might be an issue that it takes a long time until certain analysis methods become so well understood and so easy that lots of PhD students can do something and get enough research they can write papers about. It was the same story in the mid 1990s with protocols. Once there were tools, you could specify the protocol, throw it into a tool and see what comes out.

Reply: Yes, that's familiarity again, that's when you learn the system and then you attack it, and you're protected until then.

Bruce Christianson: I think it's an interesting observation that breaks on the BAN logic started after the first proof that it was correct. A cynic would say, well OK the countermeasure is obvious, just don't prove that the product is

secure. But the thing about a proof is it tells the attacker where to look because the proof holds under certain assumptions, so to find the attack I've got to break one of the assumptions.

Reply: And the same with hackers, because hackers always succeed by violating assumptions. So it's almost exactly the same model you would see with someone attacking software, saying do it again, go around the system, or the same with the exploit Wednesday, you're looking at the patch to see where you should focus your attack. Same model.

Bruce Christianson: Yes, picking what to change.

Reply: Well one thing we found is that old attacks come around again, old attacks are always new again. I have a friend, his name is Mike, and Mike was laughing a few months ago, and the reason he was laughing was he said it was an attack he hadn't seen since the 90s, but it was working again because of something somebody had changed. They never go away, they just evolve or recycle.

Saša Radomirović: Don't you have that also with infectious diseases, you have a while before they actually smoke.

Reply: Smallpox is back again, Polio is back again in the US, yes.

Bruce Christianson: Yes, even a very small population smouldering away below the inoculation threshold is enough to break out again soon.

Reply: They worried about those parents that won't inoculate.

Bruce Christianson: That's exactly it, yes.