The Economics of Cybersecurity — Lecture 5 Notes

Adam Hastings

January 23, 2024

Homework Review

Can someone volunteer to share what they found in their homework?

Really straightforward. I just wanted to gently force you to engage with what the security community does to deal with patching. The goal was hopefully for you to better understand patching to help you make sense of the assigned paper.

Discussion: General Economics of Patching

What are the economic forces that affect the ecosystem of patching?

- Who benefits from patching?
- Who loses out from patching?
- What are the costs of patching? Who pays them?
- What do we think about the "Ship it tomorrow and fix in the next release" line from Anderson's paper?
- What does a typical patch look like? How many lines of code is it?
- What are some deployment considerations when it comes to patching?
  - How easy is it to patch software that you wrote yourself on your own system? Even this may not be simple
- Why might someone *not* want to patch a system?
  - May be because they're an IT manager who is busy with other tasks
  - May be because patching a system will cause unacceptable downtime
  - May be because system is remote or unaccessible (e.g. pacemaker).

    * This brings up another question: Should companies be forced to make their systems patchable?
    * What happens if someone discovers an issue with pacemaker code? Could be a security issue or simply a functional one.
    * Healthcare devices are pretty strictly regulated by the FDA. Should other domains be regulated the same amount? Should in-home security cameras be required to have a patching system in place? What about a children's toys?
    * Side note: These questions are being asked and discussed by governments around the world today, and with some regulations (in Europe mostly) already affecting product vendors.
    * There are lots of very difficult questions to answer here that intersect with computer science, economics, policy, and law. We will discuss these topics more later in the semester.

- Are all systems patchable?
  - We already mentioned pacemakers, which are physically inaccessible. Or are they? Should we allow over-the-air software updates?
  - Some devices (particularly embedded devices) have things like security fuses which are physical fuses that are deliberately blown prior to deployment that physically remove the ability to interface with a device. For example JTAG is a set of communication protocols for interfacing between a circuitboard and a testing device. Could allow access to sensitive information on a device. You can use JTAG to interface with a board but then might blow the fuses on the device to make sure that no one else has low-level access to your device after deployment.
  - Patchability also depends on physical components. Some types of memory (like EPROM) require the use of strong UV lights to program (specifically to erase). If your program is stored in EPROM, are you going to be able to patch it?
  - In domains where performance is super critical, you may not use a general purpose processor and instead make a dedicated silicon chip (ASIC) to compute some function. If there's a security flaw in the circuit, can this easily be patched? No. Hence some domains may need to use FPGAs (essentially re-programmable circuits) which can be "patched" like traditional software.

- **Important Question:** Does giving products the ability to be updated increase or decrease the product's security level?
  - Increase: If vulnerabilities are found, they can be fixed
  - Decrease: It alerts attackers that a certain piece of software is vulnerable, and even tells you exactly what and where the issue is!
  - Decrease: It means that someone somewhere has the ability to change the code that's running on a device. Is the update mechanism secure?
    * I did a hardware security internship at Bloomberg, who makes their own biometric authentication devices (fingerprint swipe). One of the projects I worked on was writing firmware that decided whether or not to allow a patch. A lot of steps need to happen to make sure this is done securely!

  - It's not immediately clear if patching is always better. I suspect it is (and suspect that the majority of security professionals would agree with me) but this is based on intuition rather than evidence.

- How can you get everyone to patch when they need to?
  - via culture ("Patch Tuesday", second Tuesday of each month. But this leads to "Exploit Wednesday"!)
  - Auto-updates (iOS does this). Maybe appropriate for consumer devices. But will e.g. a bank want Microsoft to be able to remotely access and modify their machines? Probably not. What if the patch breaks something important?

- What might be done to make sure that patches can be done securely?
  - Depends on the domain. But generally is going to involve some level of certificates.
  - Security I was listed as a prerequisite for this class. Can someone tell me what a certificate is?
  - Let's back up even more: Can someone tell me what a digital signature is?
    - It's a set of algorithms:
    - Key generation: Create a public-private keypair. Based on special properties of fields usually.
    - Signing: Using the private key, create a signature. In ECDSA (common signature scheme), this is 64-bytes.
    - Verification: Using a hash of the signed data, the public key (available to everyone), and the signature, verify that the signature was created using the public key.
  - Conclusion: If I'm a product vendor, I could sign some piece of data (like a patch) using my private key, and then using my public key, you (or your device) could verify the signature before deciding to accept the patch.
  - What's the problem here?
    - The problem is that someone could impersonate me. A malicious attacker could make their own keypair, give you a patch with a signature, and ask you to verify the signature. And the signature verification would succeed! So we need some trusted way of verifying that the public key belongs to a specific person or company. This is where certificate come into play.
  - What's a certificate?
    - A certificate is a signature by a trusted party that a certain public key belongs to a certain individual or company.
    - Who here has used cerificates before? Trick question—all of you! Whenever you access an HTTPS website, it means that communication beween you and the website is encrypted using public key cryptography. But to ensure that you are actually communicating with the website you think you're communicating with, your browser will check the website's certificate, which says "this website's public key is XYZ" and will be signed by a certificate authority, which are companies that are trusted just to
  - What's the problem here? It's requires you to trust the cerificate authority to do the vetting process for you. This process has been compromised before!

  - If you are the device manufacturer though, you may be able to act as your own Certificate Authority though. But this brings up other questions...
  - What happens if my private key gets stolen?
    * I'd have to create a new keypair
    * What if your device was programmed on only accept one single public key though?
- Discussion conclusion: Patching is not simple. There can be significant costs involved depending on the level of security required.

A Large-Scale Empirical Study of Security Patches

- What is the purpose of this paper?
- What is the methodology used?
- What types of patches were the authors interested in? (Security patches)
- But how do you actually acquire a large-scale amount of data on security patches upon which you can do your empirical analysis?
  1. Obtained a list of vulnerabilities from the NVD (as CVEs).
  2. Found URLs in the CVE entries. Many conatined `*git*` in them. Likely link to a Git repo. Everybody here knows what a Git repo is right?
  3. Did anyone's homework contain a link to a Git repo?
  4. OK, so they get a git commit. How do they actually determine that this specific commit actually is related to security? Many times systems are patched due to functionality reasons. They authors use a logistic regression method based on n-grams to build a classifier.
     - What other paper have we read so far that uses logistic regression? (Mine)
     - What is an n-gram? Did anyone look it up? Because I'm guessing some of you didn't already know what this was.
     - Relies on manually annotating git commits into either "security" or "non-security" classifications.
  5. How can we evaluate the quality of their classifier?
     - They give numbers for recall (82%) and precision (91%).
     - Recall is the proportion of actual positives identified correctly: (TP) / (TP + FN)
     - Precision is the proportion of actual positives that were actually correct: (TP) / (TP + FP)
     - When building a classifier, these are at odds with one another.
     - So how good are these? I don't know. It seem OK to me. It sort of depends on community standards.
- How do the authors then extract a patch from the commit? (They compare file extensions before and after commit. Not sure how this is different from the commit itself! The authors also limit to file extensions of programming languages)

- How do the authors estimate vulnerability disclosure dates? Per-site web scrapers.
- How do they verify the results? Randomly sample 100 crawled pages (all relevant dates correctly abstracted). Reasonable?
- What are the methodological biases that are introduced as a result of their chosen methods?
  - Not everyone uses Git! And those that do may be of a certain kind of software or developer.
  - For example, the Android project uses https://android.googlesource.com/kernel/common as its git repo. The authors' methodology would have skipped this and other projects that were self hosted.
  - What about projects that are not open source? Can anything be done to collect data here? There may be a database of patches themselves but I'm unaware of any. It would be really hard to collect this data. You'd have to scour individual websites. Even then, patches themselves may be encrypted! Likely not a fruitful path. We have to be happy with what we can get sometimes.
  - Not all vulnerabilities are made public. Biased towards trends in open source community.
  - I think the authors do a good job of laying out the limitations.
- Next, the authors do data characterzation.
- Figure 3: Log-scale y-axis, but I think this trend has continued. Most of bug finding these days is done via automated bug-finding tools
- Figure 4: Uses CVSS Version 2. Homework had you use Version 3. Same scale though.
- Table 2: CWEs. Did anyone's CVE chosen for the homework have one of these CWEs?
  - Does anyone want me to clarify what some of these CWEs are? (They're also listed on MITRE's website)
  - What stands out to you?
  - Note that Buffer Overflow is the top CVE. Does anyone know what this is?
  - Did you know that Buffer Overflows were called the "vulnerability of the decade"...back in the year 2000??
  - When I say security is an economics problem, this is what I mean. We have programming languages that make it impossible to do buffer overflows. We have tools that make it impossible for your system to execute a buffer overflow. Yet year after year this is the top vulnerability. Are you seeing why when in comes to real world security the limiting factor is often not a technical one? This is exactly why this topic—this class—is so important. It doesn't matter that we know how to fix buffer overflows if we can't figure out a way to balance the costs of the defense. This is going to the topic of next week's class.
- Figure 5—this can be somewhat tricky to interpret. Can someone tell me what it means? (Solid line—CDF of all commits; dashed line—CVE commits. CVE commits are a small fraction of overall commits—makes sense.)
- Section 5: Null hypotheses and p-values. Does anyone know what these are?

  - Null hypothesis: hypothesis that there is no significant difference between two distributions
  - p-value: Probability that observed data occured given that the null hypothesis is true.
  - Look up statistical hypothesis testing if you want to learn more about this.

- Vulnerability lifetime method: What do we think? I'm not sure I understand the difference between "code base lifetime" and "window of exposure". Can anyone help me understand?
- Use `git blame` to determine most recent blame data across patch lines to estimate the date of vulnerability introduction. Thoughts?
- Results in Figure 7. Median vulnerability had a lifespan of 438 days. Anyone surprised by this? Study of Ubuntu kernel vulnerability found that the lifespan was roughly 5 years!
- Do more severe vulnerabilities have shorter lives? No correllation observed. What method was used? What is the advantage of this over say Pearson correlation? (Doesn't assume linearity)
- Table 3: Different vulnerability classes have different lifespans! Why?
- Figure 8: What is the interpretation (80% of vulnerabilities are patched the same day they are announced). Why is it not 100%? What would it take to get this to 100%?
- Note: Everything in that 20% is the danger zone—known vulnerability, no patch yet; all it takes is for an attacker to exploit the now publicly-known vulnerability and they can attack a system.
- Figure 8: Some vulnerabilities take over 1000 days to be patched. Maybe these are low-risk issues? Yes. 88% of high-severity were patched prior to announcement, compared to only 59% of low-severity vulnerabilities.
- Do security and non-security bug fixes always modify source code? Surprisingly no. Erroneous config settings, broken build dependencies, undocumented requirements.
- Figure 10: 50% are ¡ 10 LOC changes, 80% ¡ 100 LOC changes. Median was 7 LOC changes. Generally less LOC changes that bug fixes!
- Other comparisons are made to bug fixes. I didn't find these to be as interesting. Maybe you did.

That concludes our paper read-through. In hindsight, what did we think?

- Did they answer there questions well?
- Are there any other questions you wish they'd have answered using their methodology? (Personally I'm interested in the runtime overheads of patches but I don't know how I would do this using their web scraping methodology).
- Who is most affected by this paper? Who does it benefit?

How Much is Performance Worth to Users?

- Why do you think I assigned this paper to discuss on the day we talk about patching? Is this paper even about patching?
- You may find it interesting that the initial reason why we did this work was because we were interested in the effects that patching has on users. What does this paper have to do with patching?
  - My research area is in hardware security, and in hardware

(Give 15-minute conference presetation)

Post-talk discussion:

- What did we think of this work?
- (Answer any other questions students may have)
- One thing that you may find interesting is that the motivation for this

I want to subsantively discuss the *content* of this paper, but I do also want to briefly discuss the

- What do you think I did well in this presentation?
  - General lack of text
  - Use of visual metaphor
- What do you think could have been better?

Paper discussion:

- How reasonable is CPU frequency as a measure of performance?