Final Project Part 1

Stewart Johnston
CIS 127 – Intro to Information Security
NCMC
johnstons1@student.ncmich.edu

Fullsoft Incorporated recently suffered breach of data confidentiality, damaging its competitive advantage. Proprietary information was exfiltrated as a result of malware attack.

This document is an informal collection of thoughts to aid discussion in an organization meeting, to help make accessible to potential laymen the security team's expertise. Specifically, the incident itself is to be analyzed, as well as the impact it may carry.

Circumstances

Malware can spread through an number of vectors. Malware which is capable of actually executing and exfiltrating data is likely to take only a handful of forms.

To lay some groundwork terms: 1. Threat: Any action which can potentially harm the availability, integrity, or confidentiality of information. Need not be malicious, merely harmful.

1. Internal threat: A threat from inside the organization. This can be as simple as a well-meaning user doing something foolish, or as complex as a disgruntled employee acting maliciously.

   It is important that we educate users of their responsibility in avoiding this, but for morale it is likely best we avoid admonishing them to harshly or making them feel stupid. Users are easily made to feel stupid just by asking for help from IT or seeing how quickly computer professionals solve an issue they were struggling with for potentially hours. My experience is that it is best to encourage them and provide ample opportunity to overcome that feeling. Users who feel inconvenienced or patronized are likely to attempt to circumvent security measures, so a balance must be found.

2. Remote threat: A threat from outside the organization/network. Remote threats are likely to leverage whatever vulnerabilities they find, including the goodwill and cooperativeness of individuals in the organization.

Because we have limited time and resource, although we would ideally want to shore up every possible vector, we'll list those most likely to least likely.

1. Software intentionally but not maliciously installed by a user on the network. This is one of the most common threats, is an internal threat caused by well-meaning but ignorant users.

   (a) Far and away the most likely breed of malware is a trojan of some kind, or otherwise malicious software smuggled in on the back of useful software. They initiate on a network primarily through social engineering attacks, such as email-attachments. Trojans are often deployed with the intention of using them as a backdoor which tries to phone home to the attacker or which the attacker can use to log in to the machine remotely.

      - Not totally unlikely that a trojan may have been crafted specifically for us by a competing organization or criminal. Attacking by leaving infected Flash-Drives in a parking lot for the curious employee to pick up and plug in is not unheard of.

   (b) By distinction, a virus is usually less stealthy and usually more destructive in behavior, and it typically aims either to disrupt operations or make itself known. A virus will usually spread through infected files or programs, being attached to some host file, and upon running will attempt to infect other host files. Although the need to ride on the back of something innocent-looking may seem like trojan behavior, the viral, self-replicating nature is what makes it a virus, while trojans typically don't attempt this self-replicating behavior.

2. A remote attacker may have found a vulnerable internet-facing server and through it, pivoted elsewhere. They could leave a rootkit behind, allowing them return access. Without interacting with any member of staff, they may have been performing reconnaissance for some time. If our organization was the subject of corporate espionage, this is not an unlikely move.

Similar Incidents

Two similar occasions are available to dissect.

Panic

The software company Panic suffered theft of their source code. According to their blog from one of the developers, the thieves delivered their attack by disguising their software as a video-codec processing program called Handbrake.

1. Handbrake is legitimate software, but for a period of three days it was vulnerable, and during that time the software was "nagging them for some time to install an update".
2. When the program reported that an incremental update was not available, it prompted for a full download of the next version. The developer, just wanting to get on with it, didn't stop to think about why Handbrake needed elevated privileges to run, or about how sketchy the authentication dialogue seemed at the time.
3. Git credentials were stolen and used to clone several repositories of their source code. Panic was lucky in that the attacker did not get clones of every repository, largely because the attacker was guessing at their repository names.
    - *There is no reason to believe that we are this lucky.*
4. They were contacted by the thief to the effect of a ransom. "Pay us or we'll release it into the wild" is an effective summary. There is no reason to believe that once paid the thief will keep their promise.
    - These are digital assets which can be copied and redistributed easily, not a person or physical goods which must be returned.

They came to three conclusions about the likely impact, which will be discussed later.

Valve Software

Valve is a games company who suffered theft of source code for Half-Life 2. At the time of the theft, they had been forced to announce a delay of the highly anticipated game. The remote threat was not someone interested in monetary game, but a fan who had a history of cracking game licenses. It was shared online by someone with whom the thief had shared his discovery. In any case, the process involved:

1. The attacker scanned their network, finding vulnerabilities which exposed the layout and topology of the internal network.
2. Mail accounts were accessed by the attacker, in particular the Managing Director's. Valve came to know this by examining their logs.
3. Malware had potentially infected the Managing Director's machine.

4. Source code was copied and exfiltrated. Apparently only the main trunk of the code was grabbed, which was not in itself enough to make a playable executable.
5. Keyloggers were installed by exploiting a vulnerability in Outlook. The malware was a customized version of existing software, this fork designed to infect Valve.

Potential Impact

From examining similar accounts, there are a handful of things which may happen. This list is non-prioritized, since we can't say for sure what kind of entity was the origin of this threat, or to which extent information and source code was stolen. If we got lucky, then only a portion of our code was cloned, but we can't assume this.

- **Cracked versions of our software are released**, in a word: pirating. It is probable this was already happening, to be quite honest. If no other damage occurs, this would be a blessing.
- **Malware infected versions of our apps are released**. As the Panic developers noted, if this occurred because of a trojan of legitimate software, there is a high risk it opens up our end-users to this same kind of attack. We need to take steps to stem the flow and invalidate any developer certificates they may have stolen. End-users need to be made aware, and if we have any distribution points through third-parties, like Apple's App Store, Google Play, or linux community package managers, we must double down on working with them to prevent unauthorized versions of our software from being released through those means.
- **Knowledge of our program structure allows for further theft**, especially concerning if there is any user-data which can be accessed by someone who merely knows how we retrieve it. This would likely be the most damaging to our operations, our end-users, and our reputation. If we can stop this from becoming a user-data breach, then we should do so as quickly as possible. Possible steps include:
  - Rotating passwords, certificates, or anything else used to gain access to our data.
  - Migrating our data to different servers entirely.
  - Invalidate the old API for end-users to connect. This will put any users who do not regularly update their installations of our software out of service until they do, but that's a small consequence compared to user-data theft.

- **Competitors use this information to creep up on our competitive edge**. As Panic software notes, this clone that the attackers have will rapidly go out of date as we continue to develop and improve our software. A competitive edge is valuable, but if a competitor released software with our fingerprints all over it, that would be quite damning. This is also one of the less severe consequences we could face, in this professional's opinion. If this is all that happens, our best bet is to continue our development and release cycle, while dedicating some resources to shoring up our leaks and preventing this from happening again.

Countermeasures

This list is presented in a very rough order of what is easy, and high priority. It remains unnumbered because the priority, ease, and efficiency is not set in stone.

- The first and easiest thing we can put in place immediately is to keep our antiviral software up to date and to actually use it frequently. Scheduling anti-malware software to run on a daily basis during off-hours would be wise. Allowing each person to run anti-malware scans on their machine during their lunch break would help catch any threats which come in during the morning hours, when many people check their email.
- Rotating our current passwords and reviewing our password policy may create some productivity hiccups, but it is necessary. If we don't have 2 factor authentication in place for members with higher privileges, we should.
- Providing security training on a regular basis would be wise. This can either be done by delivering a series of videos from an outside source, or this can be done in-house.
  - Outsourcing the bulk of the work, at least initially, can cut down on the time our staff needs to spend on the topic, especially given that we have other security concerns to which we must attend. However, we must be careful that our choice of training does not feel patronizing or snide.
  - Doing this in-house would take away labor from other fronts, but would allow some benefits. For one, security staff tend to be enthusiastic about their topic, and if they can avoid being condescending to their peers, they can be very effective in transferring that enthusiasm. For another, having a live person allows for questions to be asked.
  - Whatever is done, it's important that members are educated on their responsibilities, and that we avoid frustrating or patronizing them. Generally, the enthusiasm for security will spike for a while after training, and will peter out over time, and some people may

not be engaged by material which feels remedial. For those reasons it is important we choose a method which has some novelty and repeatability, and we avoid the feeling of managerial meddling.

- A policy which can help prevent suffering from the hijacking of legitimate software is to establish an organization software distribution point for our internal users. The IT staff can evaluate software for any problems and deploy updates when they are determined safe to use.
  - If this proves to be out of our ability, then if software is not deploying security updates specifically, it would be wise to wait a week to see if any nastiness crops up. Most vulnerabilities are discovered within a week of a software update which creates the vulnerability, and the bleeding edge is sometimes a dangerous place to be. Security updates are the only thing which should be rushed.
  - Additionally, it would be wise to practice policy that software installation go through the IT staff. Not all software is something we as an organization need or want on our network. This may be difficult, but thinking well enough ahead of time to make requests and have those requests vetted should be within the capability of a software development company.
- Practicing a least-privileges model of security is almost always a wise thing to do. The efficacy of this will depend greatly on our making daily operations convenient to do without requiring elevated privileges. In effect: make sure every user logs into the network and into their own machines with only as many privileges as they need. BYOD can make this difficult to implement successfully, but can be done.
- Speaking of Bring Your Own Device: Full-disk encryption on machines which may travel outside our network is a good idea. If someone can get physical access to a box, the only way to keep them from owning it is to have strong protections on the entirety of its contents, without merely relying on the security provided when the OS is loaded.
- Machines storing sensitive information should be kept with both strong physical and digital security measures. Full-disk encryption, as mentioned, can do wonders, and we use it under the assumption that things will go wrong. For physical security, literally bolting things down may or may not be overkill. Such measures as chipped security cards, or other things in the categories of “things your have” or “things you are” would probably not go amiss. These generally do not impact productivity overly much.
- Careful examination of the structure and security of our internet-facing machines is warranted, here. If anything is found wanting, even if it isn’t the attack vector our thief used, it would be wise to fill the gaps.