My First Golang Project

Bouke van Laethem, KPN

aiki.go

Aiki is a Japanese martial arts principle or tactic in which the defender blends (without clashing) with the attacker[...] One applies Aiki by understanding the rhythm and intent of the attacker to find the optimal position and timing to apply a counter-technique. In Japanese Aiki is formed from two kanji:

合 *: *ai - joining**

氣 *: *ki - spirit**

Down the rabbit hole

A while ago, a colleague noticed attempts by an "Administrator" to log in to one of his computers. His computer had the Remote Desktop Protocol (*RDP*) service open to the Internet. As the name suggests, the RDP service allows computer owners to remotely use their desktop. The attacker was trying all kinds of passwords to get into the machine. My colleague scanned the attacking computer. The only service open to the Internet was RDP.

Perhaps someone was consciously attacking others from his own computer, forgetting he/she had left RDP open. But more likely the system had been compromised through RDP, infected with malware and was now being used to attack others. The attacker was trying to log in through RDP and only had RDP open itself. Suddenly something dawned on me.

When a system mindlessly does whatever an attacker wants it to do, we call it a *bot*. A bot attacking other systems has usually been compromised using a list of common or standardized usernames and passwords, called a *dictionary*. I figured that eventually this bot would try to log in to RDP using the same username and password which was used to compromise itself. That seemed like a simple, provable and specific enough hypothesis.

One small problem: RDP is a complicated protocol to mimic for research purposes. Luckily Secure Shell (*SSH*), another protocol used to remotely manage devices and computers, is not. Just like countless system administrators all over the world, I use SSH to remotely manage my Internet facing servers. A quick look at the number of failed login attempts for SSH on one of my servers was enough: Password guessing attacks against SSH are happening on a very, very large scale.

Building the test case

"How do you know I am mad?" said Alice. "You must be," said the Cat, "or you wouldn't have come here."

How do you know I am mad?

I immediately started writing a program in Python that pretended to be a SSH service. The idea was simple:

- Pretend to be a SSH service.
- Bots will try to log in with a username and password.
- Automatically try to log in to the attacking system's SSH service using the same username and password.

In pseudo-code:

```
for connection in SSHD:
  try:
    SSH.connecttoattacker('connection.remoteIP',
'connection.username','connection.password')
    print('It worked!: ', connection.remoteIP, connection.
username, connection.password)
  except:
    print('The King said gravely: "Go on till you come to
the end: then stop"')
```

Sadly, my coding flow soon ground to a halt. I have this way of running into Python threading and performance issues. Things were getting ugly fast. That is when I decided to start my first Golang project. Because as some of you may know, the Golang programming language is right up there on the hipster scale with beards, soy lattes and fixies. This should be reason enough to choose it as a programming language, but it also performs great and has awesome threading functionality. So Golang it was!

In some pseudo-code, the program to fake a SSH service does this:

```
package main
import (
  //some libraries I need to make this work
)
// create a private key used by the SSHd to encrypt
communications
func buildkeys() (priv_pem []byte) {
}
// set up non-bruteforcable account details
func unguessable() (username string, password string) {
}
// ssh client that can reuse captured usernames and passwords
func aiki(ip string, username string, password string) {
}
func main() {
  // start fake SSHd server
  config := &ssh.ServerConfig{
    },
  // connect back to anyone connecting to the fake SSHd
server
  go aiki(ip, username, password)
  }
```

In the end the module turned out to require a few more nuts and bolts. You can find the aiki.go source on https://github.com/KPN-CISO.

I deployed aiki.go on one system, quickly followed by five others in different IP ranges around the world. And then, I waited. Would I catch anything? And if so, on how many different systems? Well...

```
Days aiki.go has been running: 183
Number of attacking systems: 8986
Number of successful "aiki": 742
Top 10 of usernames and passwords used in successful
"aiki":
  220 admin:admin
  132 root:admin
   53 pi:raspberry
   34 root:root
   33 root:123456
   29 ubnt:ubnt
   23 root:welc0me
   22 root:000000
   21 root:openelec
   14 root:1234
Top 10 of most successful days:
   33 2016/12/29
   28 2016/03/06
   24 2016/12/25
   18 2016/03/15
   14 2016/12/30
   14 2016/12/28
   14 2016/03/09
   14 2016/03/04
   14 2016/01/04
   13 2016/12/27
```

It was really nice to have my hypothesis proven, but now I had some hard questions to answer. Was what I was doing legal? And even if it was okay in the eyes of (international) law: how could I use all of this ethically?

Laws

"Now, I'll manage better this time," she said to herself, and began by taking the little golden key, and unlocking the door that led into the garden.

Now, I'll manage better this time...

In the Jabberwocky world of information technology it must be hard for lawmakers to decide what constitutes a crime. To make sense of intangible acts often the tangible world is taken as a guide. So, can we find a real-life example to explain what aiki.go does? And can we use it to decide if what we are doing is legal? I think we can do both, but first I have to give a little background on how SSH works.

When you use SSH (version 2), the acts of connecting, authenticating, and actually doing something on the remote system are strictly separated into transport, authentication and connection steps. A diagram might help to explain this.

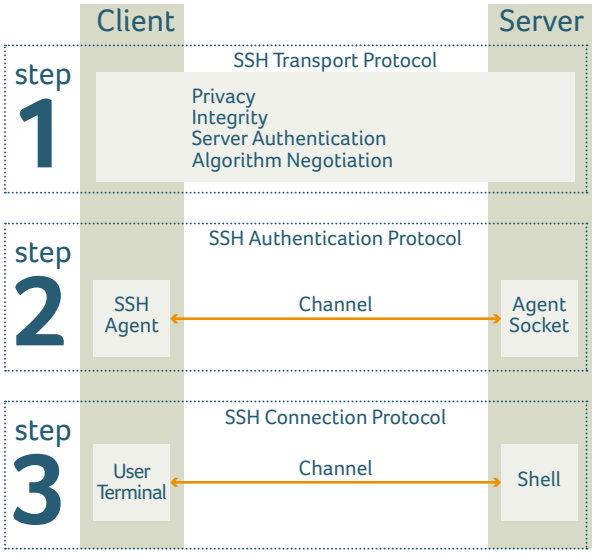

Figure 1: SSHv2 protocol

The aiki.go program works by taking the virtual key (username:password) the attacker used on us to try it on the attacking systems lock (SSH service). If the lock turns (Step 2: Authentication), aiki.go does not even bother with the digital equivalent of using the handle or pushing against the door (Step 3: Connection). aiki.go simply notes it found a working key and moves on.

I am not a lawyer but I think I am staying just on the (slightly bleeding) edge of what is legally allowed. I only try to authenticate (not log in!) with the exact username and password the attacker just tried against me. But I am not completely sure, so I sincerely hope you will let me know if you think differently. I am very much looking forward to some constructive feedback and expert opinions. But even if what I am doing is lawful, that does not necessarily make it right.

Ethics

The Queen turned crimson with fury, and, after glaring at her for a moment like a wild beast, screamed "Off with her head! Off--"

"Nonsense!" said Alice, very loudly and decidedly, and the Queen was silent.

Off with her head!

I am a hacker. I get a thrill from making things bend to my will. The buzz of getting my first positive results created a torrent of ideas. I could name and shame everyone attacking me on Twitter! I could log in to the attacking systems and remove the infection, or even destroy the remote system altogether! I could jump from system to system uprooting botnets! Off with their heads!

So I started with the first thing that came to mind and added functionality to put IP address, username and password of every successful "aiki" on Twitter. That would teach them!
Except it would not. Because there is nobody to teach. I had a look at a couple of attackers using a web browser. It quickly became clear these ruthless attackers were just cluelessly configured devices.

My sample of successful "aiki" is a collection of victims, not villains. People are usually completely unaware their devices are attacking others. Clearly they also do not know that anyone trying a couple of passwords can access their security cameras, baby monitors, backup devices, modems et cetera. If I started naming and shaming attackers on Twitter, I would just be giving access to these devices to all the Twitter trolls, violating the privacy of innocent bystanders.
As for the other ideas I had: logging in to devices to stop or attack botnets, besides being illegal, also felt wrong. Again, I would be further violating the privacy of the abused and that is not okay, ever.

Law enforcement agencies are likewise hampered in what they can do. They are usually allowed to request information about which person is behind a certain

IP address. Theoretically they could get into contact and secure digital forensic data. But that would be a lot of work and none of it would be done with the goal of helping those who have infected devices.

Conclusion

As far as aiki.go is concerned: in its current form it cannot do anything more without becoming unethical and unlawful. It has served to prove two things:

1. Most attackers out there are actually innocent victims with easily guessable passwords as their main weakness.
2. For anyone with some time and technical knowledge, it is possible take over large numbers of bots and botnets without actively attacking other systems. It would be illegal and unethical, but there are people out there who do not care about such "details". Someone might already be doing this, without us ever knowing.

In the mean time, the best response the international community has come up with is a technological arms race. Governmental and private sector defenders are out there fighting against the botnet herders. However, that fight focuses exclusively on trying to protect the (potential) victims *of* these botnets, not the victims *in* the botnets.

What I think we are lacking is the digital equivalent of the World Health Organisation (WHO) *Global Outbreak Alert and Response Network* (GOARN [1]). Perhaps Computer Security Incident Response Teams (CSIRTs [2]) could play a role in building a digital GOARN, with a focus and mandate to fight the disease by helping the victims.

In the Jabberwocky world of information technology it must be hard for lawmakers to decide what constitutes a crime.

(1) GOARN (www.who.int)

(2) CSIRTs (www.cert.org)