0x4D31 / **honeybits**

Watch ▾ 7 | ★ Star 41 | Fork 3

A simple tool to create and place breadcrumbs, honeytokens/traps or honeybits, to lead the attackers to your decoys/honeypots!

golang go honeybits honeypot honeytrap deception

9 commits | **1** branch | **0** releases | **1** contributor | MIT

Branch: **master** ▾ | **New pull request** | **Create new file** | **Upload files** | **Find file** | Clone or download ▾

0x4D31 fixed a typo | Latest commit `b2ff140` on May 1

contentgen	template for content generator added	2 months ago
docs	docs - diagram added	3 months ago
template	template for content generator added	2 months ago
.gitignore	initial commit	3 months ago
LICENSE	initial commit	3 months ago
README.md	updated the features	a month ago
hbconf.yaml	fixed a typo	a month ago
honeybits.go	template for content generator added	2 months ago

README.md

honeybits

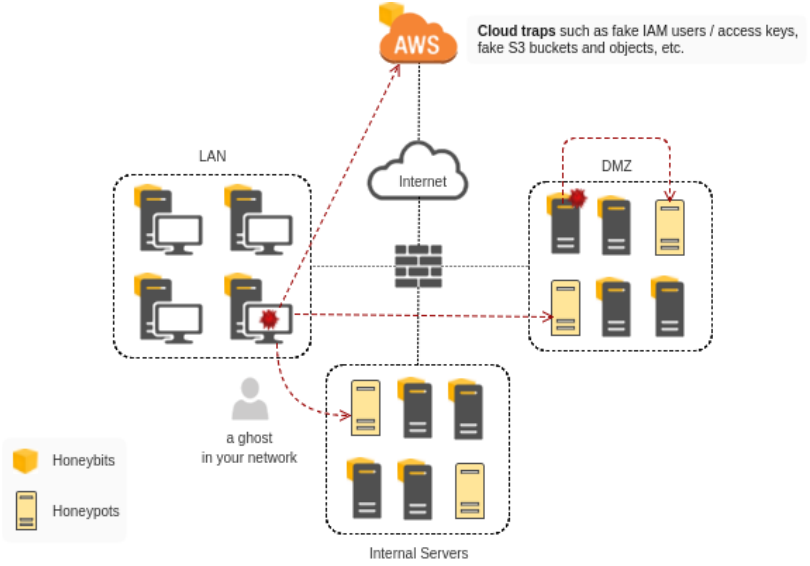
A simple tool to create and place breadcrumbs, honeytoken/traps or as I call it "honeybits", to lead the attackers to your decoys/honeypots! #cyber_deception #honeytoken

The problem with the traditional implementation of honeypots in production environments is that the bad guys can ONLY discover the honeypots by network scanning which is noisy! The only exception I can think of is Beeswarm (it intentionally leaks credentials in the network traffic and then looks for the unexpected reuse of these honey credentials).

If you take a look at the Mitre ATT&CK Matrix, you will see that 'Network Service Scanning' is only one of the many different Post-breach activities of attackers. The more you plant false or misleading information in response to the post-compromise techniques (specially the techniques under ‘credential access’, ‘Discovery’, and ‘Lateral movement’ tactics in ATT&CK matrix), the greater the chance of catching the attackers. "Honeybits" helps you automate the creation of breadcrumbs/honeytokens on your production Servers and Workstations. These honeytokens or breadcrumbs can include:

- Fake bash_history commands (such as ssh, ftp, rsync, scp, mysql, wget, awscli)
- Fake AWS credentials and config files (you required to create fake AWS IAM users with no permissions and generate access keys for them)
- Configuration, backup and connection files such as RDP and VPN
- Fake entries in hosts, ARP table, etc.
- Fake browser history, bookmarks and saved passwords
- Injected fake credentials into LSASS
- Fake registry keys

This is a small but crusial component of your deception system which should also include honeypots (ideally high-interaction ones), Log collection and analysis system, alerting, and so on.


AWS
Cloud traps such as fake IAM users / access keys, fake S3 buckets and objects, etc.
LAN
Internet
DMZ
a ghost in your network
Honeybits
Honeypots
Internal Servers

Current features:

- Creating honeyfiles and monitoring the access to these traps using go-audit or auditd
- Template based content generator for honeyfiles
- Insert different honeybits into "bash_history", including the following sample commands:
  - ssh `(sshpass -p '123456' ssh -p 2222 root@192.168.1.66)`
  - ftp `(ftp ftp://backup:b123@192.168.1.66:2121)`
  - rsync `(rsync -avz -e 'ssh -p 2222' root@192.168.1.66:/var/db/backup.tar.gz /tmp/backup.tar.gz)`
  - scp `(scp -P 2222 root@192.168.1.66:/var/db/backup.tar.gz /tmp/backup.tar.gz)`
  - mysql `(mysql -h 192.168.1.66 -P 3306 -u dbadmin -p12345 -e "show databases")`
  - wget `(wget http://192.168.1.66:8080/backup.zip)`
  - any custom commands: `(nano /tmp/backup/credentials.txt)`
  - aws

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws ec2 describe-instances --profile devops --region us-east-2
```

- Insert honeybits into AWS config and credentials file
- Insert honeybits into /etc/hosts
- Reading config from a Remote Key/Value Store such as Consul or etcd

Test:

```
$ go build
$ sudo ./honeybits

Failed reading remote config. Reading the local config file...
Local config file loaded.

[failed] honeyfile already exists at this path: /tmp/secret.txt
[done] go-audit rule for /home/test/secret.txt is added
[done] honeyfile is created (/home/test/secret.txt)
[done] go-audit rule for /opt/secret.txt is added
[done] sshpass honeybit is inserted
[done] wget honeybit is inserted
[done] ftp honeybit is inserted
[done] rsync honeybit is inserted
[done] scp honeybit is inserted
[done] mysql honeybit is inserted
[failed] aws honeybit already exists
[done] hostsconf honeybit is inserted
[done] awsconf honeybit is inserted
[done] awscred honeybit is inserted
[done] custom honeybit is inserted
```

TODO:

- Content generator for honeyfiles and file honeybits
    - note: honeyfiles are fake monitored files with random content (doesn't matter), but file honeybits are like connection, config, or backup files that may contain credentials and point the attackers to our honeypots/decoys
- Add more Credential Traps
    - Configuration, connection and backup files (file honeybit)
- Add more Network Traps
    - Monitoring some network traps using go-audit
- Add Application Traps
- Add Windows support (current version supports Linux and Mac OS X)
    - New traps including CMD/PowerShell commands history, Browser history, Saved passwords, Registry keys, Credentials, Connection and configuration files such as .rdp and etc.
- Documentation