To prove that a distributed installation of TPOT is possible a set of scripts and configurations were created to simulate the second general architecture (without the reverse proxies). These scripts and configurations makes it possible to:

- Setup the whole logging server with;
    - Elasticsearch, Logstash and Kibana
    - Elastalert
    - SearchGuard for Elasticsearch and Kibana
    - All needed certificates using your own configuration needs
    - Cockpit
- Setup the honeypots with;
    - Filebeat
    - Additional logging capabilities

The only thing you need to do is:

- Change the IP's and SSH accounts in the scripts and configs
- Generate the certificates and your own SSH key pairs
- Execute two commands that will do the rest for you on an unlimited amount of honeypot servers and a single logging server

Besides that it is possible to manage the honeypot and logging servers using the individual methods to update all the relevant configurations. Using these methods you can e.g. deploy ad hoc honeypots with ease. To monitor all the servers and manage all the containers, Cockpit comes into play. Using Cockpit you can access and monitor everything you would want to monitor. Added a new honeypot to Cockpit is as easy as providing the IP:PORT. Using authorized key access which has already been automatically setup by the script you don't even have to enter the password.

To successfully use these scripts, there are the following requirements that must be met:

Logging server:

- Has SSH key pair for root user located in /root/.ssh/*
- Has a SSH server (apt install ssh)
- Has the following system requirements:
    - At least 6GB of ram, preferably more (8GB+)
    - At least 128GB SSD, depends on the amount of honeypots how fast it'll fill up but is a good starting point.

Honeypots:

- Has TPOT version 18.11 - Sensor Installation already installed (can be automated too - https://github.com/dtag-dev-sec/tpotce#post-install-auto)
- Has the following system requirements:
    - At least 6GB of ram, preferably more (8GB is enough)
    - At least 128GB SSD, this is enough for the standard log retention period of 30 days.

Computer you are using:

- Password manager e.g. KeePass
- Isolated and clean, can be a VM
- Connected to a trusted network
- Has SSH key pair for root user located in /root/.ssh/*
- Has Fabric installed as well as Python2.7

These scripts are however not production ready and should be considered unstable and in alpha phase. Only tested with Ubuntu Server 16.04+. They do however provide a solid basis if {COMPANY} decides to use this and proves that a distributed TPOT setup not only is possible, but is also quite easy to manage and setup.

1. Maintenance and documentation

All the files that are needed for the distributed TPOT Sensor Installation are as listed below.

```
./
|-- Honeypotservers
|   |-- filebeat.yml – Custom Filebeat configuration file
|   |-- logging.sh – Custom script to configure additional logging
|   |-- tpot.service – Modified TPOT service file (NOTE: only version 18.11!)
|   |-- tpot.yml – Standard TPOT configuration file (NOTE: only version 18.11!)
|   `-- updateip.sh – Modified TPOT script to get relevant IP’s
|-- Loggingserver
|   |-- elasticsearch.yml – Modified Elasticsearch configuration file
|   |-- elk.service – Custom Logging server service file
|   |-- kibana – Modified Kibana initialization file
|   |-- Kibana objects – import these in Kibana for 210+ visuals and dashboards
|   |   |-- index.json – Modified index
|   |   |-- search.json – Modified search index
|   |   `-- visual_dashboard.json – Modified visuals and dashboards
|   |-- kibana.yml – Modified Kibana configuration file
|   |-- logstash.conf – Modified Logstash configuration file
|   |-- start_elkdocker.sh – Custom startup script for logging server
|   |-- start.sh – Modified startup script for sebp/elk image
|   `-- stop_elkdocker.sh - Custom shutdown script for logging server
`-- Scripts
    |-- create_certificates.sh – Custom script to generate all relevant certificates
    |-- fabfile.py – Custom and main script that is the heart of this PoC
    `-- fabfile.pyc – Related to fabfile.py

4 directories, 19 files
```

These files are modified from the original TPOT repository or are created from scratch and therefore called custom. All these files are changeable to your needs but are preconfigured to work out-of-the-box. The upcoming points will guide you through all of these files and how to use them accordingly. Besides these points, the scripts are also documented with comments inside. Before proceeding you need to configure the fabfile correctly with all the relevant IP’s and SSH accounts.

2. How to setup the logging server

For the logging server it is highly advised to use Ubuntu Server 18.04 LTS since that OS has been tested with the scripts and configurations. Firstly you must edit the create_certificates.sh file and execute it. While executing you need to create a CA and provide the password each time to sign all certificates. It is also advised to check all the configuration files to verify if they fit your specific needs. All configurations will work out-of-the-box if they remain unchanged, so be very careful when committing changes.

Besides that you will need a SSH key pair inside your /root/.ssh/* directory. You can generate a SSH key pair with the following command (you MUST run this command as root and do not change the standard output maps):

```
1. ssh-keygen -t rsa -b 4096
```

The method has 18 parameters that will need to be passed:

- update_sh - This refers to the update_ip.sh file **path**
- kibana - This refers to the kibana file **path**
- logstashconfig - This refers to the logstash.conf file **path**
- kibanaconfig - This refers to the kibana.yml file **path**
- elasticconfig - This refers to the elasticsearch.yml file **path**
- elkscript_start - This refers to the start_elkdocker.sh file **path**
- elkscript_stop - This refers to the stop_elkdocker.sh file **path**
- elkservice - This refers to the elk.service file **path**
- cockpit_crt - This refers to the cockpit.crt file **path**
- cockpit_key - This refers to the cockpit.key file **path**
- elastic_crt - This refers to the elastic.crt file **path**
- elastic_key - This refers to the elastic.key file **path**
- logstash_crt - This refers to the logstash.crt file **path**
- logstash_key - This refers to the logstash.key file **path**
- kibana_crt - This refers to the kibana.crt file **path**
- kibana_key - This refers to the kibana.key file **path**
- root_crt - This refers to the rootCA.crt file **path**
- start - This refers to the start.sh file **path**

You execute the method by executing via the following syntax: “fab initialize_loggingserver:parameter1,parameter2,parameter3,etc”.

3. How to setup the honeypot servers

It is highly advised to install TPOT on an Ubuntu Server 18.04 LTS since that OS is officially supported and all testing has been done on that OS. The installation can be done automatically[1] using the scripts provided by dtag-dev-sec but can also be done manually[2] if preferred.

Once TPOT and the logging server are installed it is time to execute the initialize_honeypot method in the same folder where the fabfile is located. You will need a SSH key pair inside your /root/.ssh/* directory and the other certificates/keys that you generated when initializing the logging server before executing.

[1] https://github.com/dtag-dev-sec/tpotce#post-install-auto

[2] https://github.com/dtag-dev-sec/tpotce#post-install-user

You can generate a SSH key pair with the following command (you MUST run this command as root and do not change the standard output maps):

```
1. ssh-keygen -t rsa -b 4096
```

The method has 8 parameters that will need to be passed:

- logging - This refers to the logging.sh file **path**
- filebeat_config - This refers to the filebeat.yml file **path**
- tpot_service - This refers to the tpot.yml file **path**
- tpot_service - This refers to the tpot.service file **path**
- filebeat_crt - This refers to the filebeat.crt file **path**
- filebeat_key - This refers to the filebeat.key file **path**
- root_crt - This refers to the rootCA.crt file **path**
- path - This refers to the id_rsa.pub of your machine file **path**
- cockpit_ssh - This refers to the id_rsa_cockpit.pub file **path** (This public key can be found in the /tmp folder after execution of initialize_loggingserver.)

You execute the method by executing via the following syntax: “fab initialize_honeypot:parameter1,parameter2,parameter3,etc”.

4. How to add an ad hoc honeypot to all honeypots

When adding a new honeypot it is advised to always first do this inside a test environment before you break all the honeypots. This guide is only for adding a honeypot as a Docker image, which is highly advised to do. The honeypot you want to add obviously has some ports to listen on. You must first check if there is no port conflict with other honeypots that are already running. You can check using the underneath commando’s as root depending on your TPOT version on one of your TPOT instances. You could do this by logging into Cockpit, accessing one of the honeypots from there and open a webssh session.

For TPOT == 18.11

This command will show all the listening ports. Note that ports 64xxx is management related.

```
1. /opt/tpot/bin/rules.sh /opt/tpot/etc/tpot.yml set
```

If you want to verify which honeypot is binding with which port:

```
1. cat /opt/tpot/etc/tpot.yml
```

For TPOT == 17.10

This command will show all the iptables that sets listening ports. Note that ports 64xxx is management related.

```
5. cat /etc/systemd/system/tpot.service
```

If you want to verify which honeypot is binding with which port:

```
1. cat /opt/tpot/etc/tpot.yml
```

The result of this check will either be that there are port conflicts or there aren't. If there are no port conflicts you can skip the next few sections and go straight ahead to adding the honeypot. If there is indeed a port conflict you have a decision to make. The easiest solution is to just disable the honeypot but you could also, if for example the honeypot only interferes with one port and the other ports are fine, limit the existing honeypot by removing that port but still let it be able to catch threat intel on the ports that are not in conflict.

6. Disable a honeypot (or partially by only disabling a few ports) on all honeypots

For TPOT == 18.11

Edit the tpot.yml that is inside this toolkit by removing the whole honeypot section that you want to remove. Also remove the network in the top of the configuration file. Once that is done you can upload it using update_tpotconfig via this command:

```
1. fab update_tpotconfig:PATH
```

Afterwards you need to restart TPOT using this command:

```
1. fab restart_tpot_service
```

That's it! If you're sharp you'll notice that you didn't change anything in the Logstash and Filebeat config, that's correct. Although it might seem sloppy it does not cause conflicts/unnecessary performance overhead not changing these configs and makes it easier to enable the honeypot in the future, if wanted.

For TPOT == 17.10

You will firstly need to download the correct tpot.yml (the config in the toolkit is 18.11 only) from one of your TPOT instances and remove the ewsposter[3] honeypot for privacy reasons, there is currently no method for this but you could create one quite easily yourself. Once that is done you can remove the whole honeypot section that you want to remove. Also remove the network in the top of the configuration file. Once that is done you can upload it using update_tpotconfig via this command:

```
1. fab update_tpotconfig:PATH
```

You will secondly need to download the correct tpot.service (the service in the toolkit is 18.11 only) from one of your TPOT instances, there is currently no method for this but you could create one quite easily yourself. It is needed to change the iptables that are configured inside the tpot.service file. Remove the ports corresponding with the honeypot you want to remove and save it. Then also upload it using update_tpotservice via this command:

```
1. fab update_tpotservice:PATH
```

Afterwards you need to restart TPOT using this command:

```
1. fab restart_tpot_service
```

[3] https://github.com/dtag-dev-sec/tpotce#community-data-submission

Once the honeypot that is conflicting with your new honeypot is properly modified or unless that wasn’t necessary you can add the new honeypot.

7. Adding a honeypot on all honeypots

For TPOT == 18.11

Edit the tpot.yml that is inside this toolkit by using the Elasticpot configuration as your template and changing it to your own honeypot configuration. The configuration is something that is different for each honeypot you want to add, check out the other already configured honeypots to check if you need to configure additional parameters. Also don't forget to add an additional network interface at the top of the configuration file. You can also check out an example guide on how to add Sshesame as appendix "V. Example add ad hoc honeypot TPOT". Note that this is for TPOT 17.11 but it is still useful for educational purposes.

Once that is done you can upload it using update_tpotconfig via this command:

```
1. fab update_tpotconfig:PATH
```

Afterwards you need to restart TPOT using this command:

```
1. fab restart_tpot_service
```

Now the honeypot should be up and running, verify this. If the honeypot is working as expected and is generating logs that are accessible via the host system we can configure Logstash.

Logstash is responsible for filtering all the logs it receives from all the Filebeat instances. You can configure Logstash to parse the incoming data so that you can use it properly in Kibana via Elasticsearch. You need to use the logstash.conf that is inside the toolkit.

If the logs are in JSON format, you will only need to add a filter to verify if the timestamp is ISO8601 compliant. If not, you can change that too with Logstash. Besides that, there are a set of standard variables that are the same for all honeypots like dest_port, dest_ip, src_port and src_ip (You can check that in Kibana via Index Patterns). If these parameters are not called that, you need to change them too. You can check the current Logstash config on how to do that properly for your own honeypot.

If the logs are not in JSON format, you need to create a GROK filter. You can do that using the online tool Grok Debugger (Grok Debugger, n.d.). Create the filter and give correct labels that are in line with the Elasticsearch index (You can check that in Kibana via Index Patterns). Once the Logstash filter is created, you need to test it. If the results are positive you can deploy it to all sensors using:

```
1. fab update_logstashconfig:PATH
```

You will also need to change the filebeat.yml that is inside the toolkit. Filebeat is responsible for sending the logs to the central server. Adding a new log source is quite easy. If the logs are in JSON format you can just copy the Suricata config and adjust that to your needs and if the logs are not in JSON format you can copy the Syslog config and change that. Once that is done and tested you can upload it to all honeypots using:

```
1. fab update_filebeatconfig:PATH
```

Now everything *should* be working. It is possible you made a configuration mistake so always verify the input in Kibana thoroughly.

For TPOT == 17.10

The process for lower versions is the same except that you can't use the toolkit logstash.conf file. You can get that online and fix it by replacing [type] for [fields][type] and replacing the whole input and output section by copying it from the logstash.conf from the toolkit. The rest of the process is completely the same.

8. Other tips and tricks for maintenance and deployment

The fabfile contains the following methods:**Error! Not a valid link.**

All the get, restart and update methods are useful for management and updating of the honeypots. You can also add methods yourself as they are straight-forward to add. The reverse proxy methods are not yet finished due to time limitations but can be used for future work.

Depending on the specific threat intel you want to collect it is advised to change all the honeypots that are running on the TPOT servers. There are a lot of very obvious detection methods that are indeed being used by attackers to determine if they are inside a honeypot. So if you don't change them you will observe detection methods resulting in crashes of the honeypot or weird responses letting the attackers know it is not a real session. Besides that you should think about changing the standard TPOT management ports or apply strict IP whitelisting (which you should) to prevent easy fingerprinting of honeypot servers by more advanced attackers.

Furthermore it is advised to create Elastalert rules to be notified when commands are being executed on the servers, when there is a sudden increase in events or there are no events coming in at all.

If the infrastructure is compromised it is advised to wipe all the servers and reinstalling the whole infrastructure, which is not as much work as it sounds since all the up-to-date configuration files are already on you management laptop.

The toolkit is currently only supporting the TPOT Sensor Installation type. If you want the NextGen or Collector installation that is also possible. As the Sensor Installation already contains all the honeypots you can disable the honeypots that are not in line with the desired installation type by modifying the config files as explained in the guide in this sub question.