Skynet design document v0.1.093012

Overview

This system is designed to automate the process of scanning specified subnets on a scheduled basis. Data from each scan is stored for historical purposes allowing the administrator to identify change over time. Automated reporting can be used to provide regular updates on security status as well as alert the administrator when anomalies are detected. Scans are compared against a previous “baseline” scan, defined for each scanning host.

This is a “living” document and is not feature complete. Check back often for updated versions.

General Features / Components

Cloud component

- NMAP
- Local timing and parameters data
- Local control daemon

Control system

- Control nmap options ?
- Identify IP blocks (CIDR notation)
- Control scan time
- Can we feed back into this?
- Push parameters to cloud
- Should be able to define multiple destinations
- Retrieve finished data from the cloud

- Create ndiff files?

Visualization system

- MySQL database of ndiff data?
    - Might be overkill
- Text-based ndiff data?
    - Easy to keep this encrypted
- Combination of both?
    - MySQL for metadata
    - Text for full scan data
- PHP graphing of changes
    - Time per scan
    - Number of hosts per scan
    - Number of ports per scan
    - Change in hosts/ports per scan

Reporting subsystem

- Automated reporting
    - Reports triggered by control daemon
- Multiple report types
    - Pre-defined
    - Custom

- Configurable email address
- non-emailed reports?
    - On the fly reports
    - GUI only reports

Component Detail

Cloud Component

The “cloud” piece of this software is the dumb workhorse piece of the system. Setup should be minimal and easily deployed on disparate systems.

Instructions are delivered via flat text files placed into a configuration directory. The local system parses these files and builds a localized timing table for spawning processes. This localized data is stored in a flat text file with a predefined format. Something similar to a cron table seems appropriate. The parsing system should create a hash table of existing configuration files to identify what is new and what can be removed from the timing table. (Should this data be encrypted? What advantage does this give an attacker?)

A spawning daemon is responsible for reading the timing table and spawning new processes at the appropriate time. New scans are spawned as separate processes with their PID being noted by the spawning daemon. The spawning daemon should identify if the previous scan process has completed prior to starting a new process. In the event of an existing process, the daemon should identify if the process is still running (PID file and PID exists) or if it died in process (PID file only). It should send an appropriate notification to the administrator identifying the problem for dead PIDs and only send a notification for existing PIDs if an override flag is not set. In the case of a dead PID, the new scan should be spawned as requested. If the PID still exists, however, the spawning daemon should only spawn the process if there’s an override flag set. This gives the administrator control to run scans on a tighter schedule when the run-time of a single scan may exceed the period of time between scans.

Finished scans should encrypt the scan results using a public GPG key and the plain text version of the file should be scrubbed. (Can we encrypt on the fly as the scan is running?) All completed files are stored in a holding area until the central processing

system retrieves them. After retrieval, reports are scrubbed from the system.

Timing Table Format

minute hour day month override_flag ip_range nmap_options

Control System

The control system is the central brain of the scanning system. It is responsible for interacting with the administrator, pushing schedule data to the scanning systems, and retrieving scan data from the scanning systems.

The control system stores all schedule data in a MySQL database. New schedule configurations are pushed out on a manual basis as the administrator chooses. Data is pushed to the remote systems via a simple SCP process. Automated retrieval of data occurs on a timed schedule. Retrieved data should be decrypted using a private GPG key. Metadata from the scans is added to a MySQL database and the raw scan data is stored in a predefined directory structure. Files should be named appropriately to indicate date and time of scan so manual interaction with files is simplified.

Visualization System

The visualation system is essentially the GUI front end for the system. It allows administrators GUI access to the control system for scheduling scans and reports as well as setting a new baseline. It also provides a visualization of the data being reported from the scanning systems. The visualization system provides graphical views of data such as actve hosts, active ports, filtered ports, and average scan times. The graphical system will use the metadata stored in the MySQL database to generate these graphs.

Possible graphs : * Number of hosts per scan * Number of ports per scan * Time per scan * Changes per scan

Reporting Subsystem

Pre-defined automated reports can be scheduled withing the reporting subsystem. Reports are associated with one or more timed scans and run on a predefined schedule. Results from the scans are summarized and presented in an easily

digestible format. Reports can be delivered via email, or viewed via CLI or GUI output. Reports should be cached for a period of time allowing quick retrieval of past reports.

Reports should include various statistics relevant to the scanning. The time taken for the scan, number of hosts and ports identified, new ports found, old ports removed.

Back End design

MySQL Database Definition

Spawner table

- id int
- server_ip int
- subnet int
- mask int
- nmap_options text

Timing table

- id int
- scan_time timestamp
- spawn_id int