Enhancing OS vulnerability scanners:
from a single box to hardened multi-node scan clusters

Protect your information assets with real-time threat detection.
Introduction

- Developer, consultant, SaaS architect, DevOps lead @ SAP
Introduction

• Co-founder of Binary Confidence
Introduction

- Developer, consultant, SaaS architect, DevOps lead @ SAP
- Co-founder of Binary Confidence:
  - Expert Consultancy
  - Trainings and live simulations
  - MSSP
  - Security Operations Centre (SOC)
  - Emergency Response Team
MSSP Encryption box

- Log & data collection
- Infra. and self monitoring
- Patch management
- Vulnerability scanning
The Challenge

Guys, we need to automate our network scanning! Are you in?

- Critical infrastructure
- Several datacenters
- Hundred(s) VLANs
- Thousands devices
- Air-gapped
- ..yet cost effective
The Options

- Greenbone / OpenVAS
- Nessus
- Rapid7
- Qualys
The Options

- Greenbone / OpenVAS
- Nessus
- Rapid7
- Qualys
What we got?

• Open Source w. community updates
• Web UI - GSA
• API and CLI – OpenVAS Manager
• Scalability ↔ ↑
• Master supports 15+ slaves and 150+ tasks
• Configurability
• Multiple output formats (PDF, HTML, CSV, XML)
• Reporting incl. Σ and Δ
Reported data

High (CVSS: 7.5) 80/tcp
NVT: phpinfo() output accessible (OID: 1.3.6.1.4.1.25623.1.0.11229)

Summary
Many PHP installation tutorials instruct the user to create a file called phpinfo.php or similar containing the phpinfo() statement. Such a file is often times left in webserver directory after completion.

Vulnerability Detection Result
The following files are calling the function phpinfo() which disclose potentially sensitive information to the remote attacker: http://metasploitable/phpinfo.php

Impact
Some of the information that can be gathered from this file includes: The username of the user who installed php, if they are a SUDO user, the IP address of the host, the web server version, the system version(unix / linux), and the root directory of the web server.

Solution type: Workaround
Delete them or restrict access to the listened files.

Vulnerability Detection Method
Details: phpinfo() output accessible (OID: 1.3.6.1.4.1.25623.1.0.11229)
What else do we need?

OK, we’ve got the foundation, what else do we need?

1. Fast installation
What else do we need?

OK, we’ve got the foundation, what else do we need?

1. Fast installation and final deployment
2. Running
   a. Reconfigurations
   b. Security Monitoring
   c. Operation Monitoring
3. Air-gap support - updates
4. Simple and safe HL communication
5. Backups and High-availability
6. Hardening
OpenVAS
The Ingredients

**Master on Ubuntu 16.04**
- OpenVAS 9 GSA
- OpenVAS 9 Manager
- OpenVAS 9 Scanner
- SSHD for tunneling
- Zabbix 3.0 server&agent
- Salt 2017.7 master&minion
- OS updates repo (HTTP)
- OpenVAS 9 repo (RSYNC)
- OSSEC / Logstash / (ELK)

**Slaves on Ubuntu 16.04**
- OpenVAS 9 Manager
- OpenVAS 9 Scanner
- AutoSSH for tunneling
- Zabbix 3.0 agent
- Salt 2017.7 minion
- Rsyslog / Beats
HW Requirements

- Mini PCs: 1..2 LAN ports
- 1U servers: 6..10 LAN ports

Master
- Single master / HA
- For 15 slaves: 2 cores, 4GB RAM, 128GB disk, no scanning

Slave
- Value: 1 core, 2GB RAM, 32GB disk
- Optimal: 2-4 cores, 4GB RAM, 64GB disk
Communications

• External O->M communication does support 2FA
  – OpenVAS GSA&Zabbix: TCP 443 O->M and SSH: TCP 22 O->M

• All M<->S communication tunneled - autoSSH
  – OpenVAS scanner: TCP 9390 M->S
  – Zabbix monitoring: TCP 10050 M->S
  – Salt remote execution: TCP 4505, 4506 S->M
  – OpenVAS RSYNC: TCP 873 S->M
  – OS & services updates: TCP 80 S->M

• Approx. data transfer:
  – Idle: M->S: 60 kbps, S->M: 80kbps
  – Scan: M->S: 100 kbps, S->M: 100kbps
  – Update: M->S: megabytes for a weekly update
Deployment & add scanner

- From Sources vs. Packages vs. Upgrades
- SVN -> GitHub: [https://github.com/greenbone](https://github.com/greenbone)

- CA Certificate of slave: /var/lib/openvas/CA/
- Create a user on slave: openvasmd --user=creds01
OpenVAS tools: CLI/Python/Dialog

- GitHub: [https://github.com/greenbone/gvm-tools](https://github.com/greenbone/gvm-tools)
  - gvn-cli – XML
  - gvm-pyshell – Python3
  - Even on Windows: gvm-cli.exe & gvm-pyshell.exe

- Other interesting projects:
  - [https://github.com/mikesplain/openvas-docker](https://github.com/mikesplain/openvas-docker)
  - [https://www.seccubus.com/](https://www.seccubus.com/)
Automation

- New slave deployment:
  - USB key w preseeded Ubuntu Server
  - MAC 2 hostname&IP
  - Run Salt-minion
- Update packages
- Update deployment
- Routine maintenance
Monitoring

- OS, basic/added services, ports and updates
- Utilization – don’t overutilize existing infrastructure
- Master-Slave connectivity
- OpenVAS services and ports
- Service status
- Tasks and results
- Update status and timestamps
- Negative checks
- Reporting to operators
Under development

- Automated delta reports
- Auto ticket creation for critical/high vulnerabilities
- Findings to Elastic (https://github.com/austin-taylor/VulnWhisperer)
- Master HA
- Cluster basic auto healing
Don’t forget about these

- Make sure everyone knows
- Adjust your monitoring
- Brute force / Default creds?

- Hardening
- Work instructions
- False sense of security

- Scheduling / utilization:
  - Lines M<->S, S<->T
  - Master, Scanner or Targets
Takeaways

1. OpenVAS – stable and amendable foundation to start with

2. Automate everything: Preseed USB, Zabbix, Saltstack

3. Communicate to SOC, educate operators (false sense of security)
Your Private Guardians

Peter Kleinert
CTO & Co-founder

www.binaryconfidence.com
peter.kleinert@binconf.com

Protect your information assets with real-time threat detection.