Threat intelligence for makers

Mark Menkhus, CISSP, CSSLP, ITIL Expert
menkhus@icloud.com
https://github.com/menkhus/threat_inteligence_for_makers

CSTTF title

- Security Threat Intelligence for Software Builders Testers and Maintainers
- CSTTF 2015
- August 20, 2015
-
-

Audience

- People who make or support products and services.
- Intelligence staff in roles that have the opportunity to decrease threat for companies and organizations… if I only knew what to look for.
- Security researchers who want to make a deeper impact on a particular client
- Spies, espionage agents, attackers

Requirements

- Threat intelligence stems from
  - what are were protecting
  - What is our obligation
  - How do we protect?
  - Warning - this is not a magic bullet

Why Intelligence?

- Reduces cost
- Systems are not delivered in isolation
- Connectedness creates emergent properties
- Defensive postures of our customers are not enough to protect them
- Allows for focus on topics that impact the business
- Increases Response

Focus

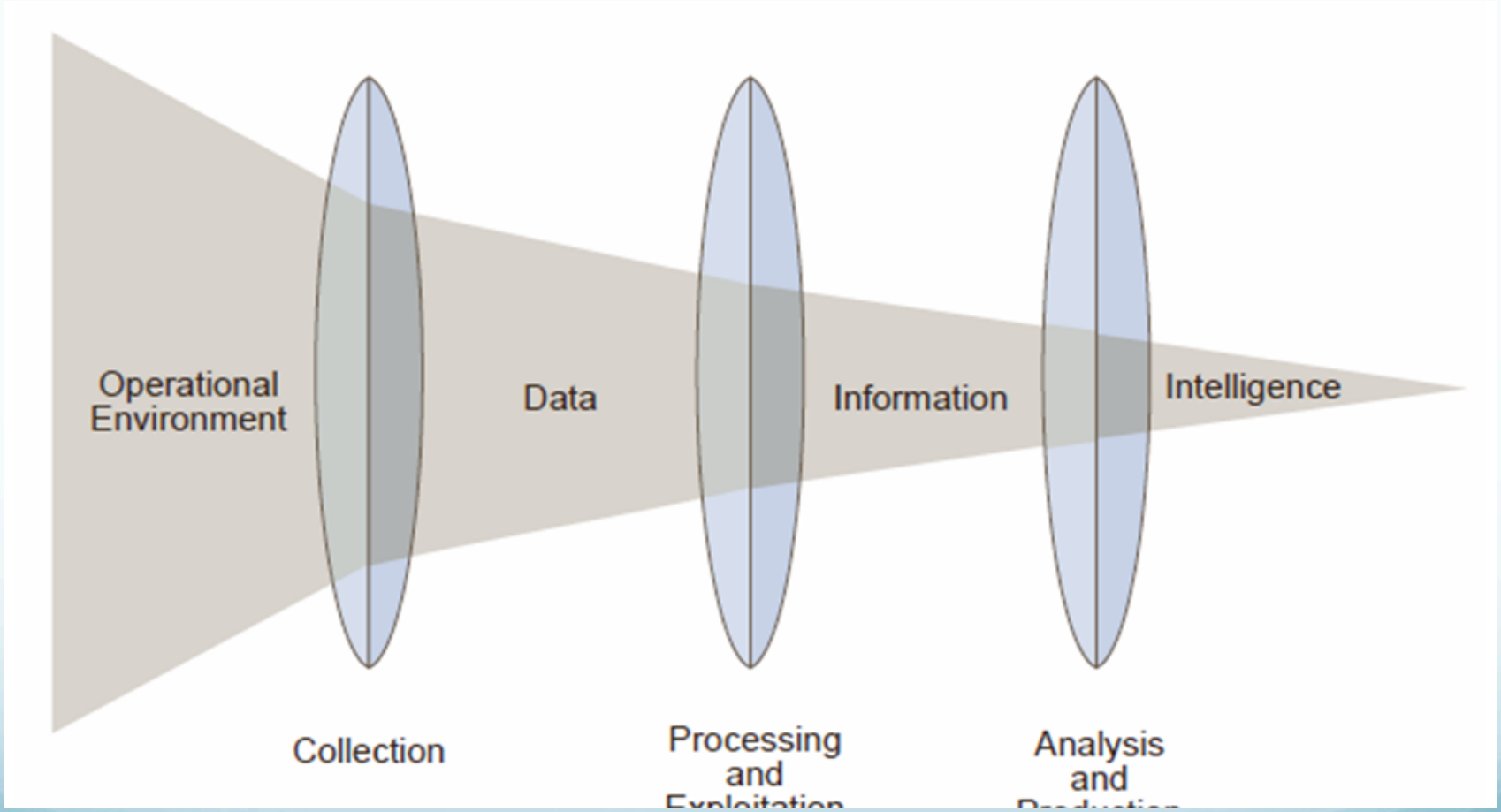
Methodology:

- Activities informed by the composition of the software projects and their environments

Intelligence approach

- Opportunistic
- Strategy
- Plan
- Scope
- Action
- Measurement

More Focus


Operational Environment
Data
Information
Intelligence
Collection
Processing and
Analysis and

Strategy

- We want to have a positive impact based on the customer experience and outcomes.
- Find threats to the systems we use to produce, support and deliver to service to the customer.
- Threats to the components we delivered to the customer.
- Threats to the software you write, your own innovations and IP, ouch!

How?

- Inform risk management approaches for the software we are building
- Think like the enemy
  - What is the easiest approach to find threats
  - What is the most effective approach to find many targets
  - Can I cheat and use Google?
  - What could Shodan do?

Think like the enemy

- What am I attacking
- What are it's weaknesses
- Could I find your weaknesses in a sea of systems?

Offense = Defense

- Compositional analysis defines the direction of search.
- Version analysis defines the shape of the horizon
- Latent bugs define the technical debt of your systems
- Risk models inform your tolerance for vulnerability
- If you are delivering general purpose software, your risk models need to be clear, and rational

Conventional Wisdom

- The more specialized your software the more you can document the risks, use cases and narrow the requirement for risk response.
- APT and insider threat suggest that this is outmoded.

Plan for technical debt

- Find known security bugs, fix during early lifecycle
- For large systems choose times when code can be updated
- What is the release vehicle for your updates?
- What is your ability to fit a fix or series of fixes into your life-cycle.
- Avoid one off patches.

Measureable outcomes

- Being Informed
- Understanding the depth of technical debt
- Plan for response
- Respond
- Inform the customers, system maintainers, and compliance staff
- Be transparent.

Before your customer's scan

- 3rd party code inventory
- Vulnerability management in your delivered product
- What services did you implement that has latent security bugs
- What services did you implement that has emergent security bugs

Sources

- News
- email lists
- Social Sites
- NVD CVE data
- Exploit databases
- Vendors
- Security vendors
- Breach notifications

You

- Your businesses as a source of information
- 3rd party software used in your code
- License analysis registries
- 3rd party code lists as part of open source behavior
- Libraries
- Code management tools
- Test tools
- Operating environment types
- Components used in system integration
- Supplier security requirements
- Hosting vendors
- SaaS used
- Tools used to produce software

Create associations

- What is the intersection between sources of information and your business
- Mash-up
- Lookup
- Set theory
- Word association
- Sounds like
- Histogram
- Discussion in the hallway
- Solicit inputs
- Scanner input
- Duck tool reports

Start Simple

- Use existing databases
- CVE from NVD
- CVE-Search
  - https://github.com/wimremes/cve-search
- Toolswatch/vFeed
  - https://github.com/toolswatch/vFeed
- What packages do you use?

Leverage

- indicators of threat - vulnerability lexicon
- Look in email for your packages
- OSS-security list
- mash-up vulnerability lexicon with open source material
- Simple database lookup

Decisions

- Do you use the vulnerable feature
- Do you use the vulnerable version
- Threat models
- network attacker
- web attacker
- reuse or replay of access controls
- escalation of access controls

Case examples

- Simple weekly standup
- Fire drill
- Registry based known bug finding
- Find 3rd party bugs in build tools
- Supplier validation

The standup

- Periodic CVE analysis, read it like a news paper
  - at least 100 CVEs per week
  - score from 0-10
  - Eliminate the noise, create a don't care list
  - Choose a level of criticality
  - use a maven, wizard, guru
  - throw away architectures, vendors not used
  - cascade and evaluate what is left

Fire drill

- Create and Use the source inventory look up in database
- Search for newest products impacted
- Try a SQL CPE match
- Manually evaluate reported versions against what is actually used
- Email a report and hold a meeting to evaluate

Automation

- Search Package Name and Version information
- Use manifests in package managers
- Leverage CPE
- Count the depth of technical debt in years
- Count the depth in number of CVEs
- Look at the severity
- Validate and file bug reports

Build Tools

- Putting Automation Intelligence into the build environment
- OWASP dependency-check - https://jeremylong.github.io/DependencyCheck/
- Falco https://github.com/menkhus/falco
- JavaScript retire.js https://github.com/victims/victims-enforcer
- Victims - https://github.com/victims

What's in a name?

- For automation use Common Platform Enumeration in your registries
- Vendor
- Package
- Version
- Use CPE database from Mitre https://nvd.nist.gov/cpe.cfm

What's in a name

- If you are lost, use a fuzzy match on the name
  - Levenshtein edit distance
    - How close is my list name to the CPE name?
  - Search on vendor names
  - Search on package names
  - Versions last
  - Use a duck tool for package names
- Validation of versions in systems
  - Distraction
  - Use the source, strings in compiled code
  - Use a duck tool and then validate

Histogram

- What to do when you have tens of thousands of 3rd party dependencies?
- Divide and conquer
- Code disclosures as part of license
- Put the work as part of enhancement or maintenance tasks
- Chew slowly

Histogram


jquery
apache-commons-logging
openssl
apache-commons-lang
apache-commons-collections
log4j
apache-commons-codec
spring
java
apache
boost
sqlite
zlib
apache-commons-httpclient
apache-commons-io
api
microsoft
perl
security
hibernate
itext
xml
slf4j
library
framework
antlr
linux
web
postgresql
windows
axis
jdom
busybox
apache-commons-dbcp
apache-commons-pool
apache-commons-net
apache-commons-cli
apache-commons-fileupload
apache-commons-beanutils
apache-commons-digester
apache-commons-configuration
apache-commons-discovery
apache-tomcat
apache-log4j
apache-cxf
apache-xerces-java-xml-parser
mysql-connector-java
jakarta-taglibs-standard
libhibernate3-java
libquartz-java
libaopalliance-java
tinyxml
python
ruby
glibc
expat
simple
toolkit
nhibernate
dom4j
snmp4j
ant
test
tomcat6
net-snmp
json
jaxen
JQuery
javassist
log4net
jfreechart
kernel
net
core
rhino
nagios
coreutils
freemarker
castor
hsqldb
jna
xalan
jstl
grails
junit
cglib
curl
algorithm
service
services
logging
jython
asm
flex
bouncycastle
xerces-c
package
wink
aop
jta
commons
wix
libxml2
atlas
guava
standard
lua
jakarta-oro
opensaml
ntp
7-zip
dhcp
ini4j
pack
jersey
common
enterprise
subversion
xstream
patch
unzip
ldapjdk
message-digest
parser
apache2
javamail
installer
google
the
diffutils
eclipse
wsdl4j

Histogram

Package	Project Instances	Vulnerabilities in CVE
Jquery	32	2
OpenSSL	21	104
SQLite	21	13
boost	13	12
antlr	11	0
fedora	1	44
Perl	10	143

Predicting risky packages

- Risk rank = [(number of instances) *1+ (# of CVEs)] /100
- Use a ceiling
- Self Leveling, things only get so bad
- Look at ranges, create a risk assessment model
  - What are the most risky and do they need special attention?

Doing this regularly

- Acquire an up to date CVE database
- Packages and versions are just key value pairs
- Search in CVEs
- CPE's point to CVEs
- Report the most recent CVE with the biggest score
- Implemented in Falco code
  http://github.com/menkhus/falco

Anecdotes – silent scanner / supplier analysis

- Gathering package names for targets is not hard
  - License adherence data ~1 hour
- Convert to CPE compatible form ~ 10 minutes
- Search in database ~3 minutes
- Validate ~4 hours
- Repeat search and Report ~30 minutes
- Significant vulnerabilities found

Findings

- Flaws found that are not seen by conventional scanner
- Log into bug tracker
- Scored
- Report on technical debt, how many flaws, not just the latest one. Could be 1,2,5,10 years from current
- Pick the most risky, not the most egregious
- Don’t forget to update your inventory

Technical debt openssl 2015

```
$ sqlite3 vfeed_db 'select distinct cveid from cve_cpe
where cpeid like "%a:openssl:openssl%"
order by cveid;' | grep CVE-2015 | wc –l
21
```

OpenSSL Technical debt

- 2015 – 21
- 2014 – 26
- 2013 – 5
- 2012 – 8
- 2011 – 12

....

In-build process security

- OWASP security check
  - Maven https://jeremylong.github.io/DependencyCheck/
- Falco
  - Developer created registry
  - Works in make and other script like environments on win,lin,mac
  - https://github.com/menkhus/falco

Open Source information

- Solicit a list of sites that serve up topical security news
- Create filters
- Report daily
- Refine process
- List of sites on this talk's github page
- List of example keywords talks github page

Example sites

- Hacker News
  - https://news.ycombinator.com/
- Slashdot
  - http://slashdot.org/
- Reddit netsec
  - https://www.reddit.com/r/netsec
- The register
  - http://theregister.co.uk

For pay services

- Data scientists
- User interfaces
- Graphs and charts
- Trending
- API's
- Feeds
- Customized reports for your needs
- Meet the need of business threat intelligence team

Making Friends/Intel network

- Threat intelligence teams in-company
- SOC/CSIRT/PSIRT teams
- Patch teams
- Customers
- Security Mavens
- ISACs
- CERTs

Conclusions

- Vulnerability management is knowledge based and intelligence approaches allow for structure and process
- No magic bullet
- Tools and data readily available
- Where does this fit in to your lifecycle
  - Does your customer assume you are doing this?
- Who does this now?
  - Cloud vendors, RedHat, Open Source software vendors

references

- Slides and notes https://github.com/menkhus/threat_inteligence_for_makers
- Duck tools
  - Palamida, Blackduck, Sonatype
- FOSS tools
  - OWASP dependency-check - https://jeremylong.github.io/DependencyCheck/
  - Falco https://github.com/menkhus/falco
  - JavaScript retire.js https://github.com/victims/victims-enforcer
  - Victims - https://github.com/victims