



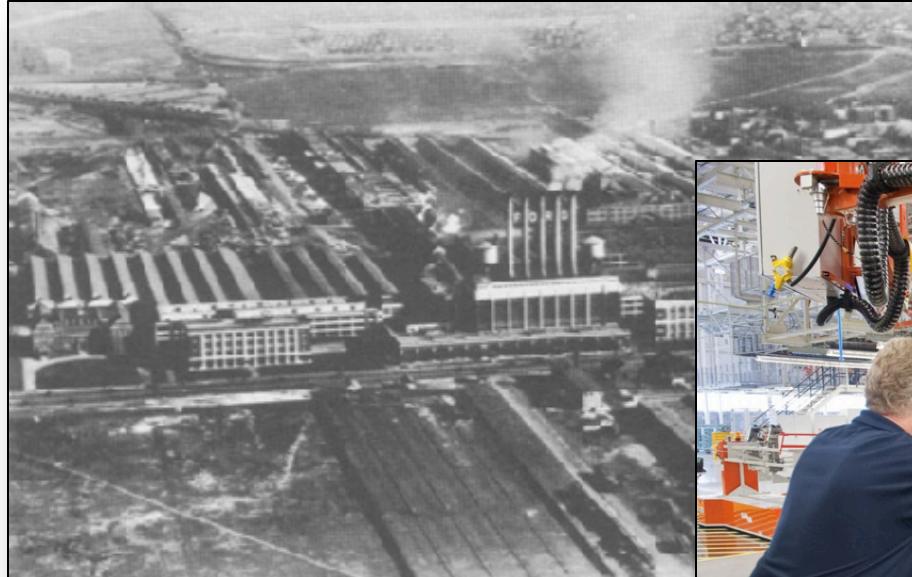
Removing the security and legal
bottlenecks in a continuous world



What can we learn from other industries?

Brian Fox @brian_fox

INDUSTRIAL EVOLUTION



```
<sourceArchive>struts-core-1.2.8-SONATYPE-001</sourceArchive>  
Presentation Layer
```

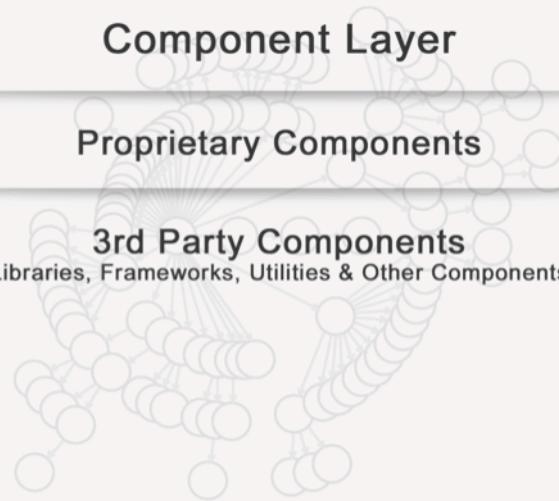
```
<sourceArchive>struts-core-1.2.8-SONATYPE-001</sourceArchive>  
Business Logic
```

Component Layer

Proprietary Components

3rd Party Components

(Libraries, Frameworks, Utilities & Other Components)



```
01010001010011101010110000101011101010101010010  
01000101010110101000100101010101010101000010101  
Database
```

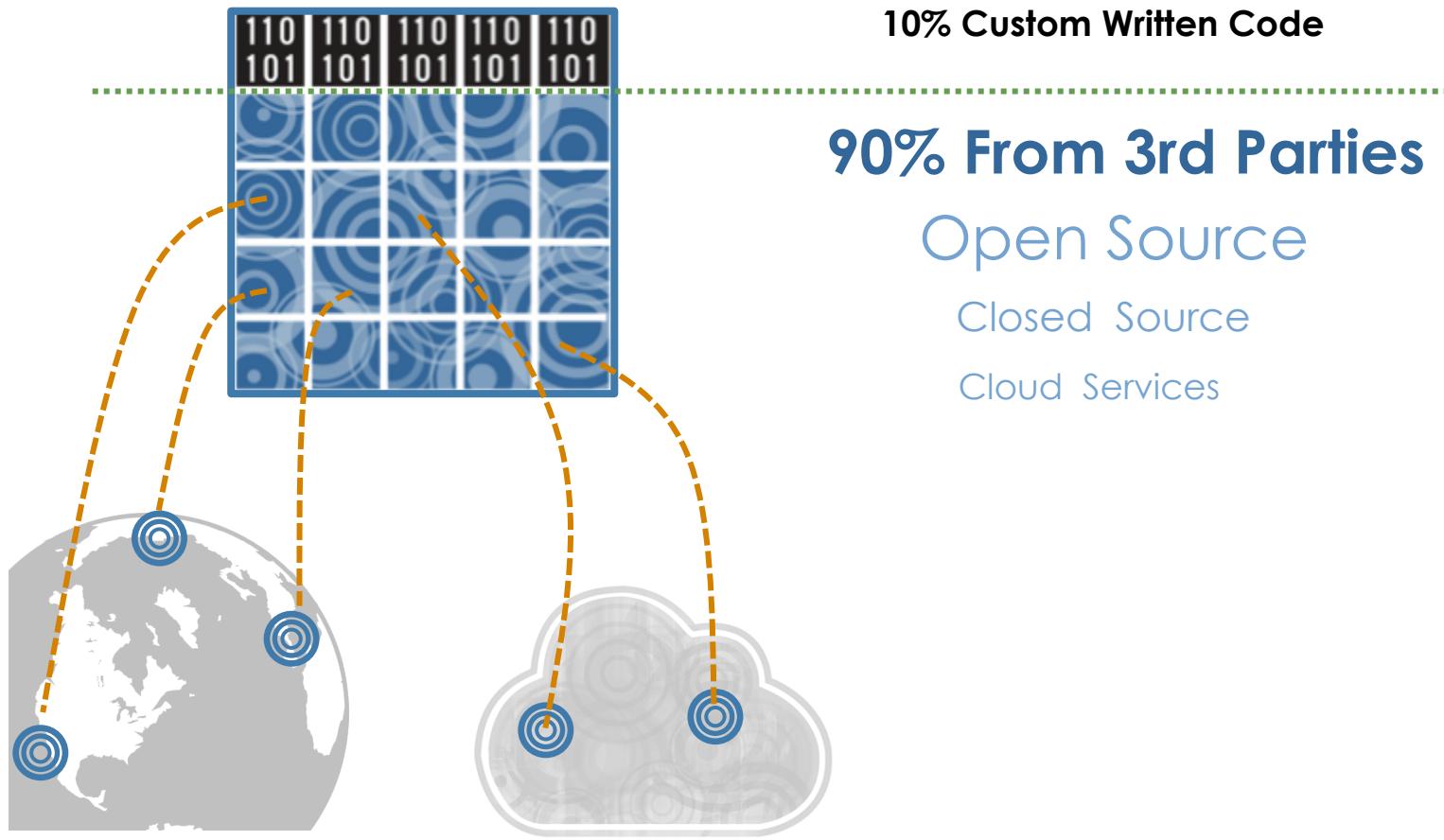
```
01010001010011101010110000101011101010101010000  
0010101011101010100010101010101010000010101  
Operating System
```

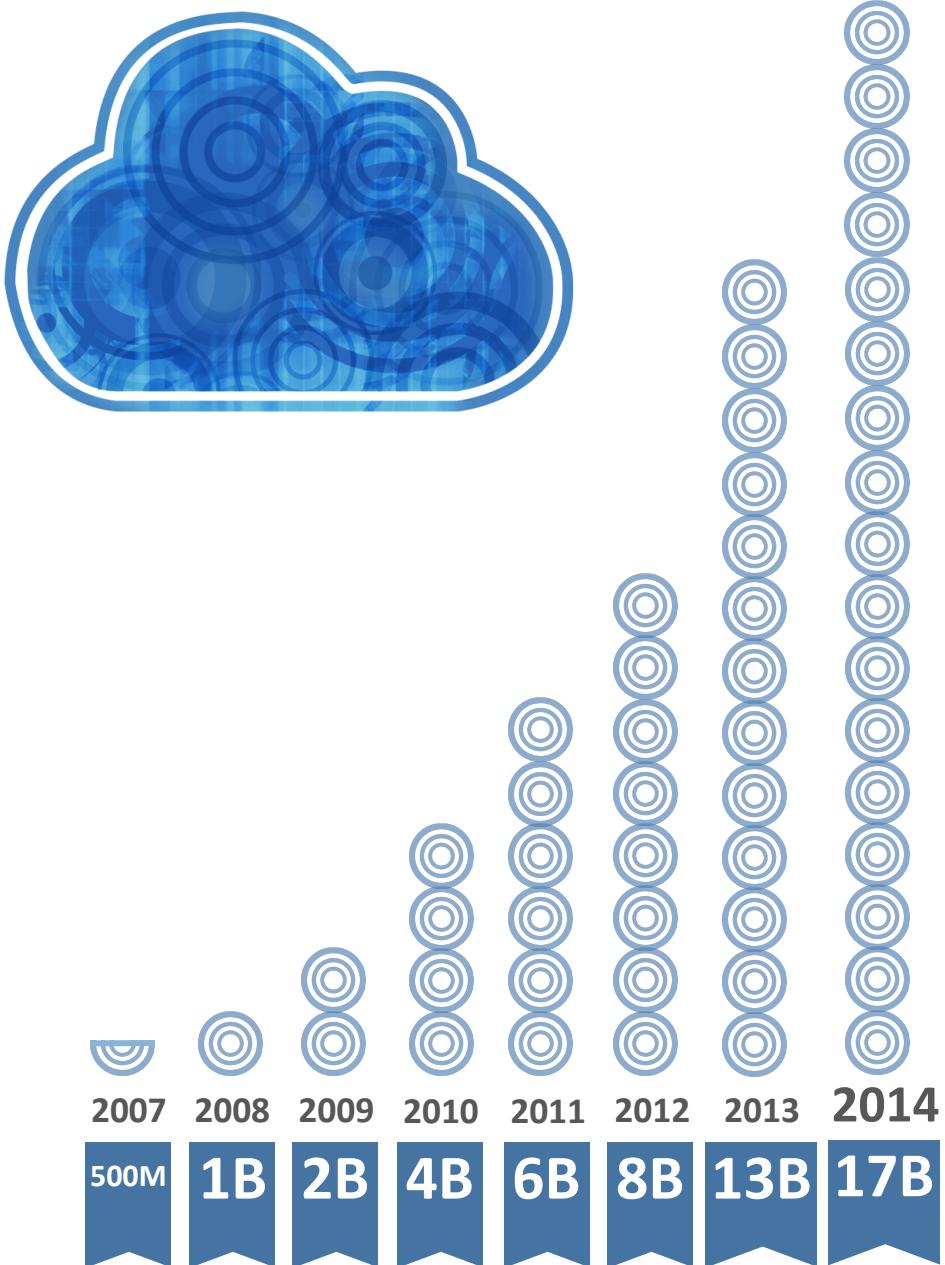
```
01010001010011101010110000101011101010101010000  
0010101011101010100010101010101010000010101  
Firmware
```

```
01010001010011101010110000101011101010101010000  
00100010101011010100101110101010101000010101  
Network
```

HOW DEPENDENT ON 3RD PARTIES ARE WE?

Typical Application





Open source usage is

EXPLODING

Yesterday's custom
code is now replaced with

OPEN SOURCE
components

ON ONE HAND WE AREN'T DOING ENOUGH

It just might get worse from here...

Security as evidence

KEY QUESTIONS

Where are **Attackers** most focused?

Where are **Defenders** most focused?

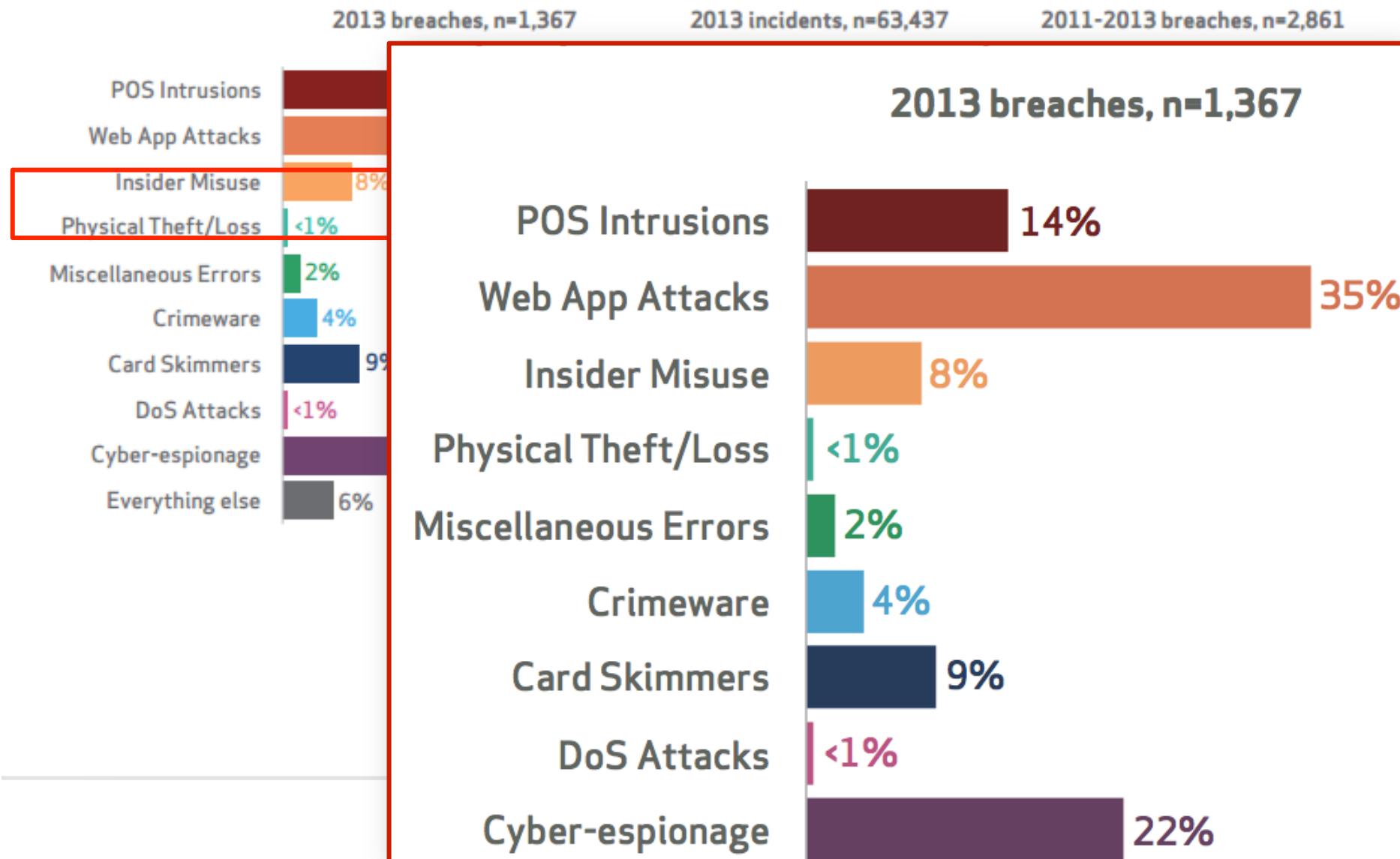
Which **Activities** have the most security impact?

MOST ATTACKED: WEAK SOFTWARE IS #1 ATTACK VECTOR

Figure 16.

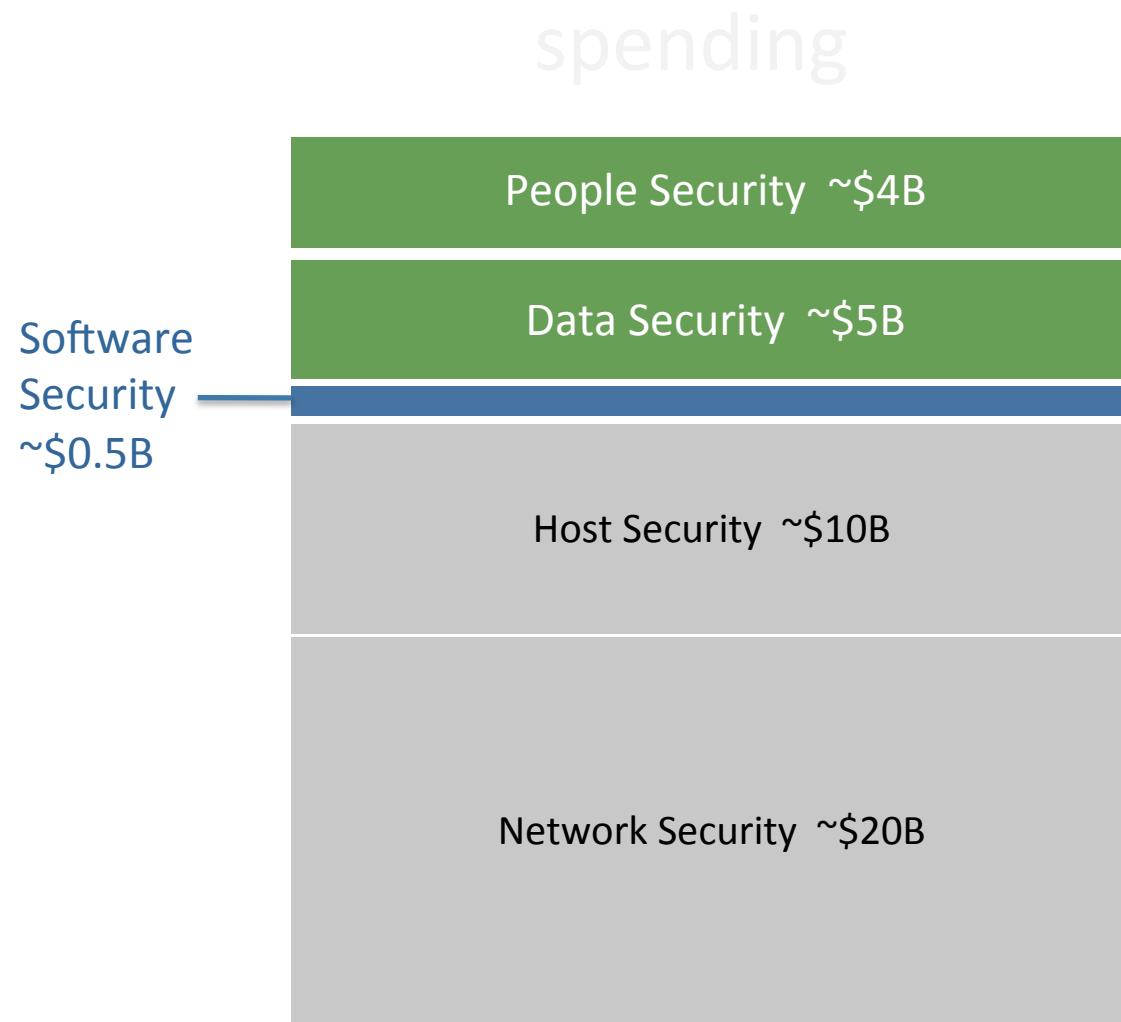
-2014 Verizon Data Breach Investigations Report

Frequency of incident classification patterns



LEAST SPENDING/PRIORITY: WEAK SOFTWARE

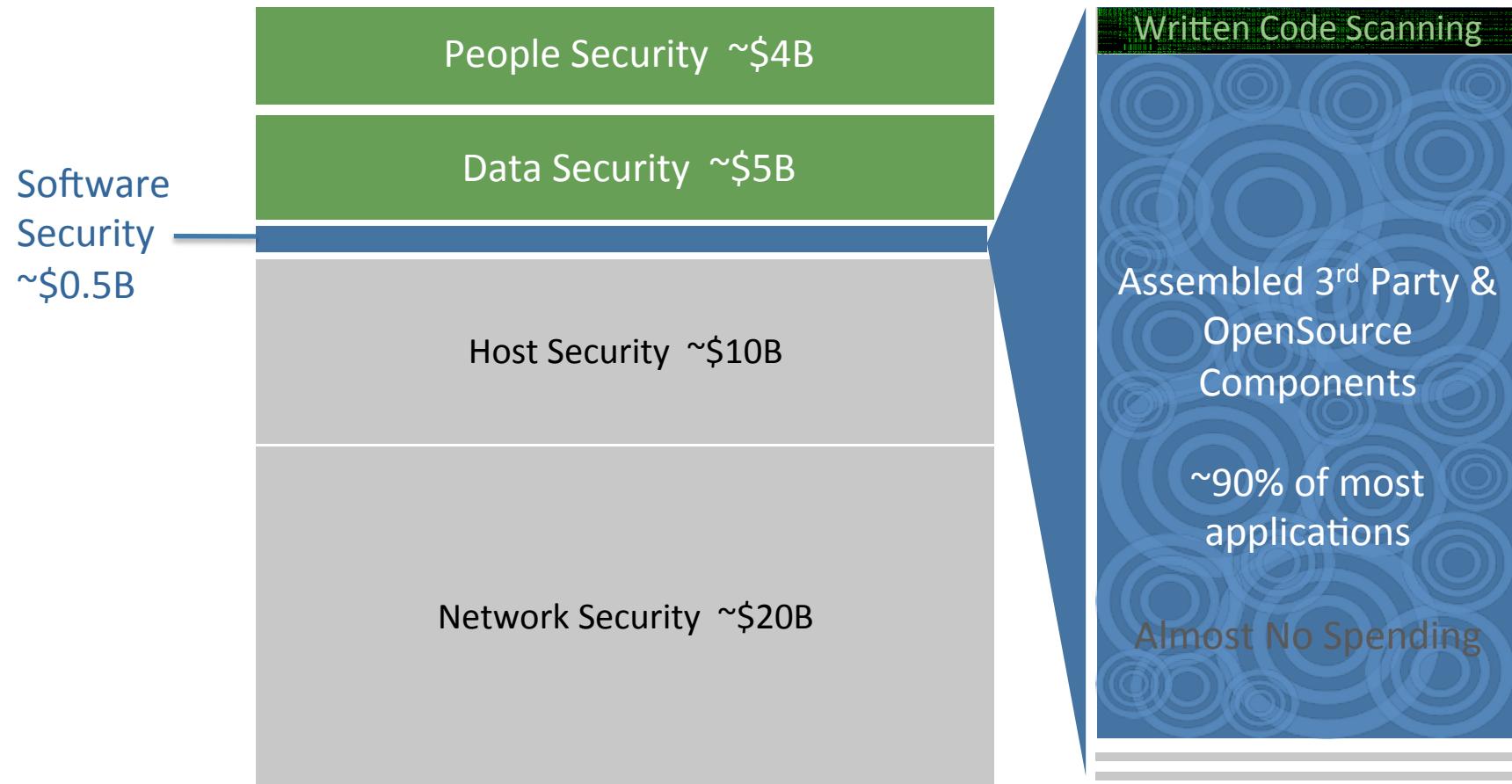
Software Security gets **LEAST** \$ but **MOST** attacker focus



LEAST SPENDING/PRIORITY: WEAK SW

Software Security gets **LEAST \$** but **MOST** attacker focus

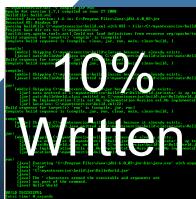
Worse, within Software, existing dollars go to the 10% written spending attack risk



MOST IMPACT: BUY/BUILD DEFENSIBLE SOFTWARE



The software & hardware we build, buy, and deploy. 90% of software is assembled from 3rd party & Open Source



THINK LIKE AN ATTACKER



When software was first being written,
finding exploitable code was like

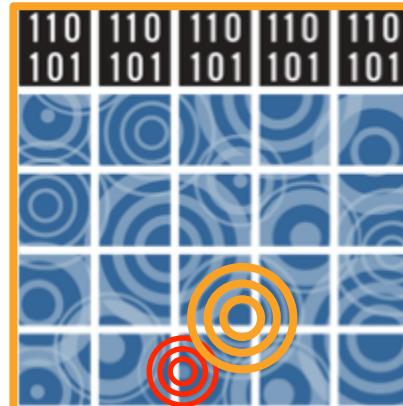
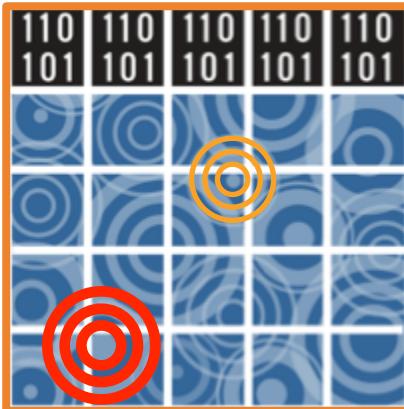
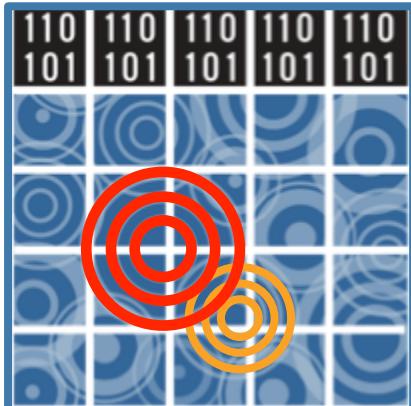
LOOKING
for a needle in a
HAYSTACK

THINK LIKE AN ATTACKER

Now that software is

ASSEMBLED...

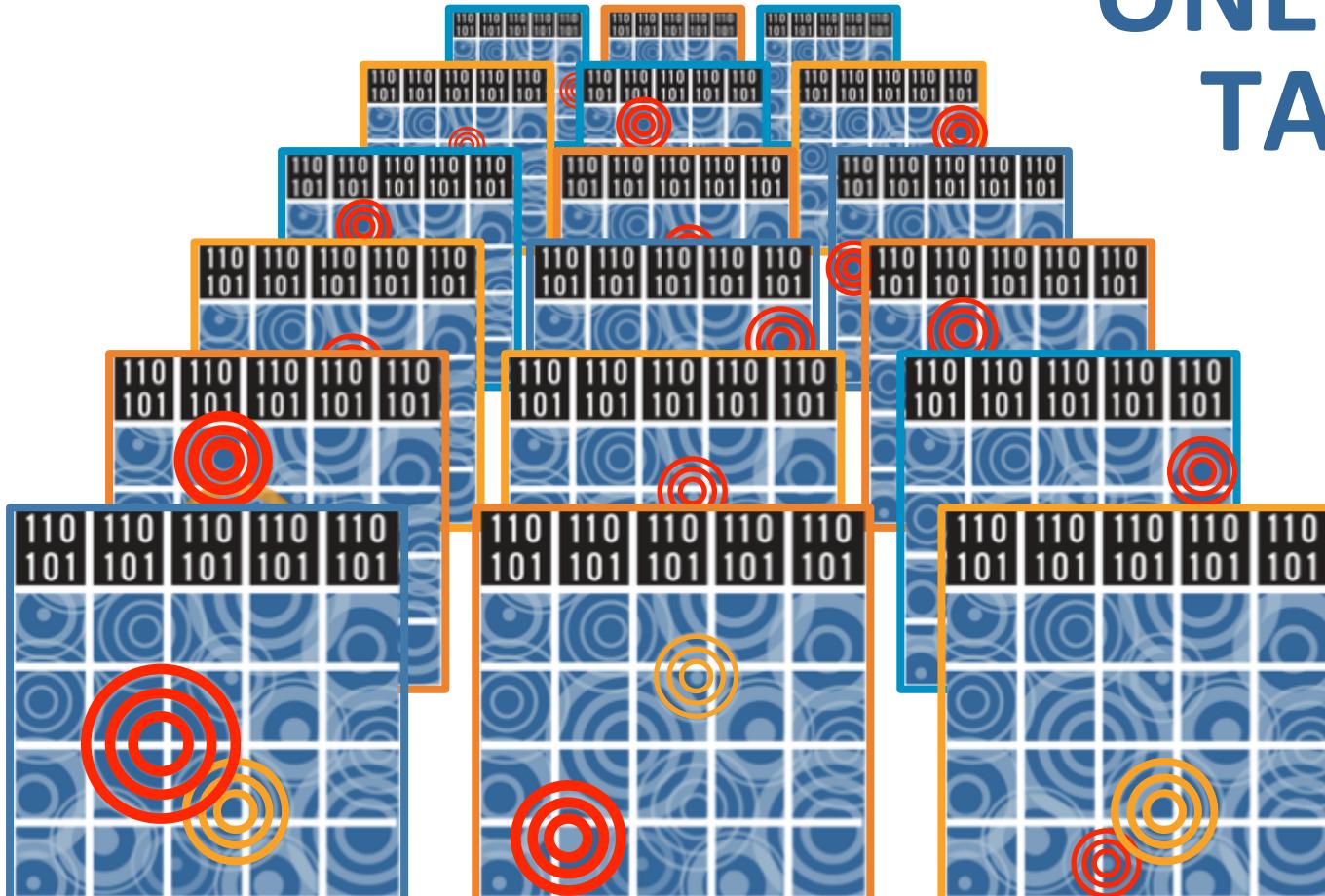
Our shared value becomes
our shared attack surface

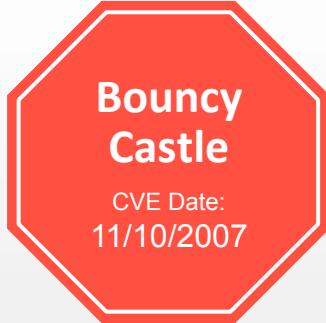


THINK LIKE AN ATTACKER

One risky component,
now affects thousands of victims

ONE EASY TARGET





Bouncy Castle

CVE Date:
11/10/2007

Java Cryptography API

CVSS v2 Base Score:

10.0 HIGH

Exploitability:

10.0

Since then

11,236

organizations
downloaded it

214,484 times



HttpClient

CVE Date:
11/04/2012

Java HTTP implementation

CVSS v2 Base Score:

5.8 MEDIUM

Exploitability:

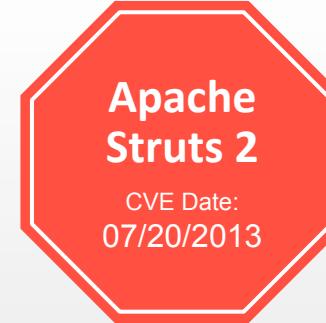
8.6

Since then

29,468

organizations
downloaded it

3,749,193 times



Apache Struts 2

CVE Date:
07/20/2013

Web application framework

CVSS v2 Base Score:

9.3 HIGH

Exploitability:

10

Since then

4,076

organizations
downloaded it

179,050 times

STRUTS

Global Bank

Software Provider

Software Provider's Customer

State University

Three-Letter Agency

Large Financial Exchange

Hundreds of Other Sites

UNCLASSIFIED

 FBI FLASH

FBI LIAISON ALERT SYSTEM
#M-000016-BT

(U) The following information was obtained through FBI investigation and is provided in conjunction with the FBI's statutory requirement to conduct victim notification as outlined in [42 USC § 10607](#).

(U) The FBI is providing the following information with high confidence.

SUMMARY

(U) Cyber actors have engaged in malicious activity against various U.S. entities. As a general matter, these actors have multiple tools at their disposal and can represent a significant threat to targeted victim organizations. Such actors have recently targeted financial and educational networks by exploiting an unpatched Apache vulnerability.

TECHNICAL DETAILS

(U) On July 16, 2013 Apache announced Struts 2 vulnerability (CVE-2013-2251 - Multiple Remote Command Execution Vulnerabilities), affecting Struts 2 versions 2.0.0 through 2.3.15. This vulnerability allows an attacker to remotely execute arbitrary Object Graph National Library (OGNL) expressions. It can be mitigated with an update patch to version 2.3.15.1.

(U) The FBI is distributing the indicators associated with these intrusions to enable network defense activities and reduce the risk of similar attacks in the future. The FBI has high confidence that these indicators were involved in the recent intrusions. The FBI recommends that your organization help victims identify and remove the malicious code.

(U) The following signatures will assist in capturing malicious activity related to the Apache Struts 2 vulnerability:

```
Alert tcp any any <-- any 80 (msg:"CVE-2013-2251_1";
content:"(new%20java.lang.ProcessBuilder|new%20java.lang.String|[]|")";)

Alert tcp any any <-- any 80 (msg:"CVE-2013-2251_2";
content:"(new+java.lang.ProcessBuilder|new+java.lang.String|[]|")");

Alert tcp any any -> any 80 (msg:
pcre:"/\\"/action|?|action|redirect|/|");
content:"(new+java.lang.ProcessBuilder|new+java.lang.String|[]|")");
```

(U) Additionally, actors have downloaded:

```
http://www.greenbuilding.or.kr/202.91.74.102/somo/rspl
http://www.qhxidi.com.cn/plus
```

Please contact the FBI
FBI CYW

2013-07-19 17:30:20 [全局] 资源工具 | 版本: struts2漏洞 struts2-exp

工具: NL_Struts2_Exploit <=2.3.1.5 cve-2013-2251
(2013-07-19 17:30:20) [全局] [漏洞利用工具]
语言: VS2010 C# (.NET Framework v2.0)
版本: 1.0
作者: NL
资源: http://qhxidi.com/nlog.html
发布: 2013/7/19 17:27:34

简介:
该脚本是针对Struts2漏洞的利用工具，可以利用Struts2的任意命令执行漏洞，从而在目标网站上执行任意的命令。该脚本使用.NET Framework 2.0编写，可以在Windows平台上运行。

<http://struts.apache.org/release/2.3.x/docs/s2-016.html>

THOUGHT LEADERS ARE TAKING ACTION

The slide is titled "OWASP Top 10 - 2013 rc1" and "The Ten Most Critical Web Application Security Risks". It features a green header with the OWASP logo and the text "RELEASE CANDIDATE FOR COMMENT". The main content is slide A9, titled "Using Components with Known Vulnerabilities".

A9 Using Components with Known Vulnerabilities

Threat Agents	Attack Vectors	Security Weakness	Technical Impacts	Business Impacts	?
?	Exploitability AVERAGE	Prevalence WIDESPREAD	Detectability DIFFICULT	Impact MODERATE	Consider what each vulnerability might mean for the business controlled by the affected application. It could be trivial or it could mean complete compromise.
Some vulnerable components (e.g., framework libraries) can be identified and exploited with automated tools, expanding the threat agent pool beyond targeted attackers to include chaotic actors.	Attacker identifies a weak component through scanning or manual analysis. They customize the exploit as needed and execute the attack. It gets more difficult if the used component is deep in the application.	Virtually every application has these issues because most development teams don't focus on ensuring their components stay up to date. In many cases, the developers don't even know all the components they are using, never mind their versions. Component dependencies make things even worse.			

Am I Vulnerable to Known Vulns?

In theory, it ought to be easy to figure out if you are currently using any vulnerable components or libraries. Unfortunately, vulnerability reports do not always specify exactly which versions of a component are vulnerable in a standard, searchable way. Further, not all libraries use an understandable version numbering system. Worst of all, not all vulnerabilities are reported to a central clearinghouse that is easy to search, although sites like [CVE](#) and [NVD](#) are becoming easier to search.

Determining if you are vulnerable requires searching these databases, as well as keeping abreast of project mailing lists and announcements for anything that might be a vulnerability. If one of your components does have a vulnerability, you should carefully evaluate whether you are actually vulnerable by checking to see if your code uses the part of the component with the vulnerability and whether the flaw could result in an impact you care about.

How Do I Prevent This?

One option is not to use components that you didn't write. But realistically, the best way to deal with this risk is to ensure that you keep your components up-to-date. Many open source projects (and other component sources) do not create vulnerability patches for old versions. Instead, most simply fix the problem in the next version.

Software projects should have a process in place to:

- 1) Identify the components and their versions you are using, including all dependencies. (e.g., the [versions](#) plugin).
- 2) Monitor the security of these components in public databases, project mailing lists, and security mailing lists, and keep them up-to-date.
- 3) Establish security policies governing component use, such as requiring certain software development practices, passing security tests, and acceptable licenses.

WHY IS THIS SO HARD?



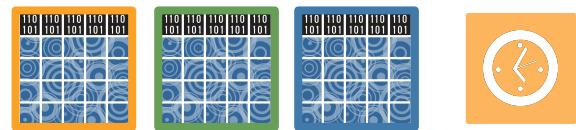
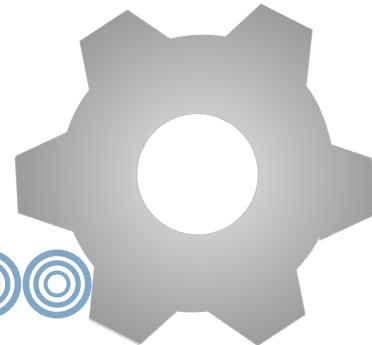
Modern software development

HAS CHANGED



Our process

HASN'T CHANGED ENOUGH



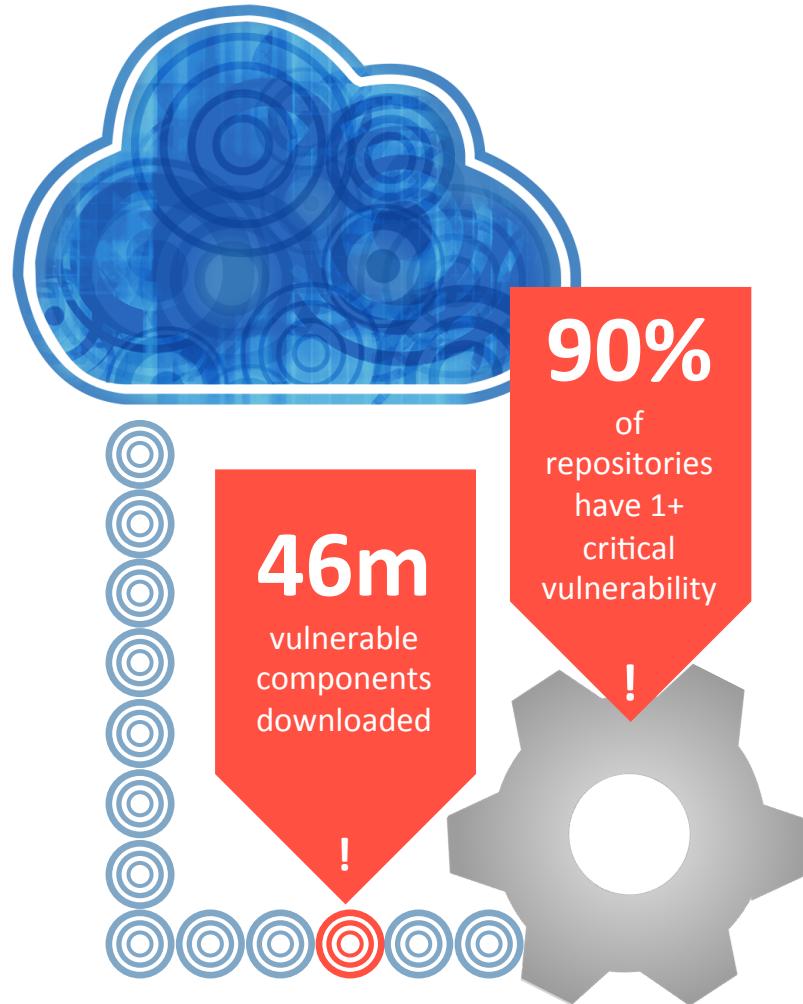
Use of components creates a
**SOFTWARE
SUPPLY CHAIN**

COMPONENT
SELECTION

DEVELOPMENT

BUILD AND DEPLOY

PRODUCTION



Today's controls

AREN'T WORKING

THE NEW LIFECYCLE

Impact on
Releases per Year
(Cycle Time)

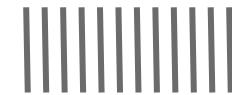
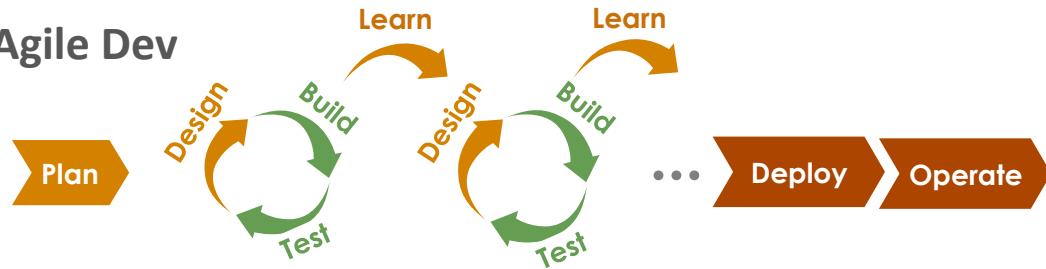
Traditional Lifecycle (Waterfall)



1-2

Cycle Time: Months-Years

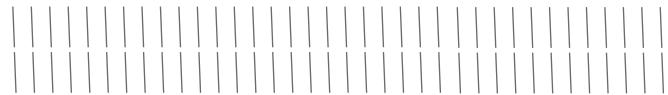
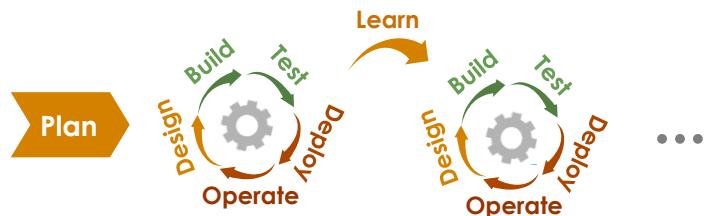
Agile Dev



10-20

Cycle Time: Days-Weeks

Modern Lifecycle (+DevOps, Continuous *)



100-200

Cycle Time: Minutes-Hours

THE NEW LIFECYCLE

Governance?

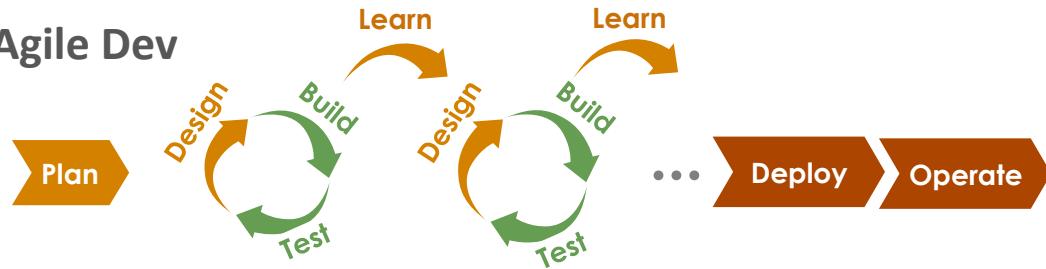
Traditional Lifecycle (Waterfall)



Manual

Cycle Time: Months-Years

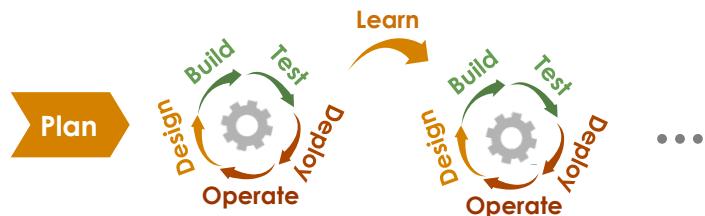
Agile Dev



Manual + Point Tools

Cycle Time: Days-Weeks

Modern Lifecycle (+DevOps, Continuous *)



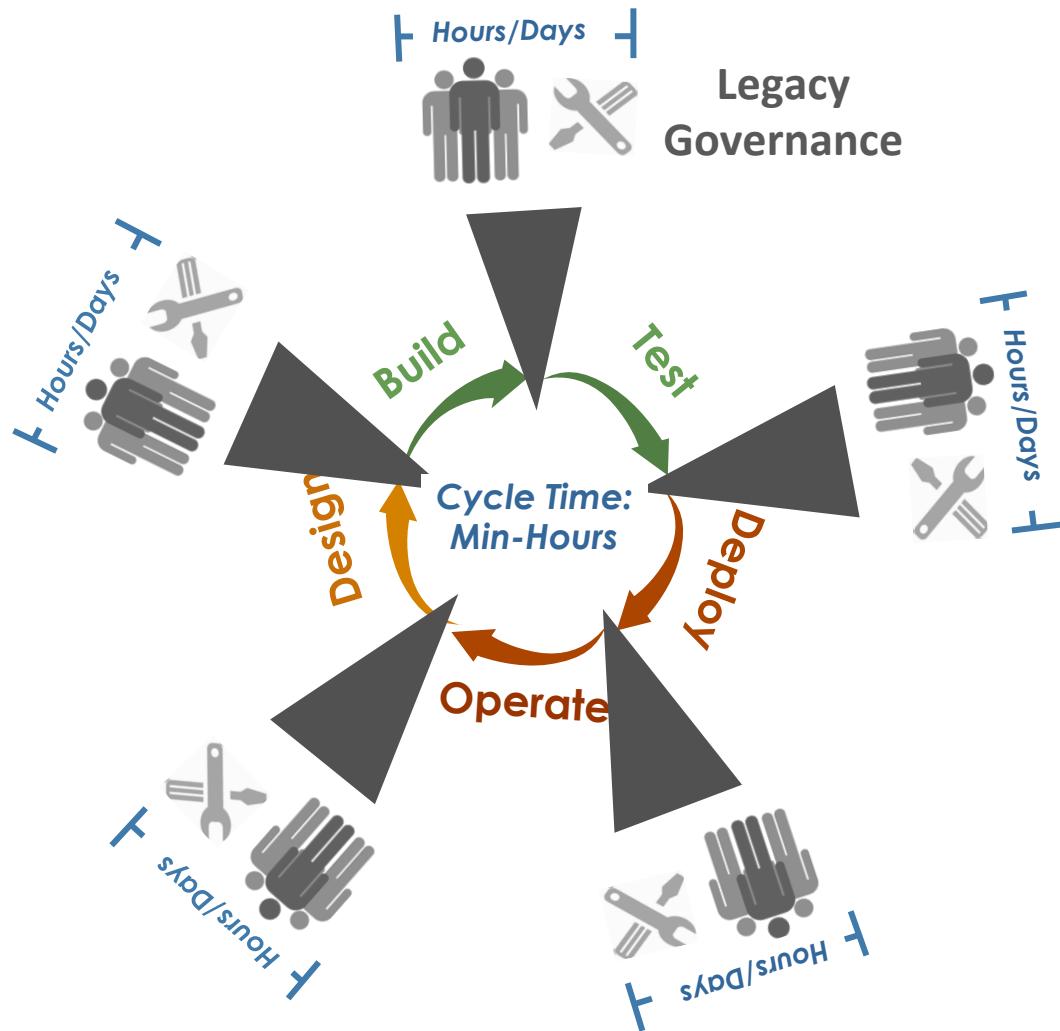
New Approach



Policy-Driven Automation

Cycle Time: Minutes-Hours

CYCLE TIME SQUEEZE



*If it does not fit,
It does not get done.*

- Work Arounds
- Batch Scans
- Rework
- Exposure

Go Fast **OR** Sleep at Night

ON THE OTHER HAND

Anti Patterns

Ask someone first

(aka hurry up and **wait**)

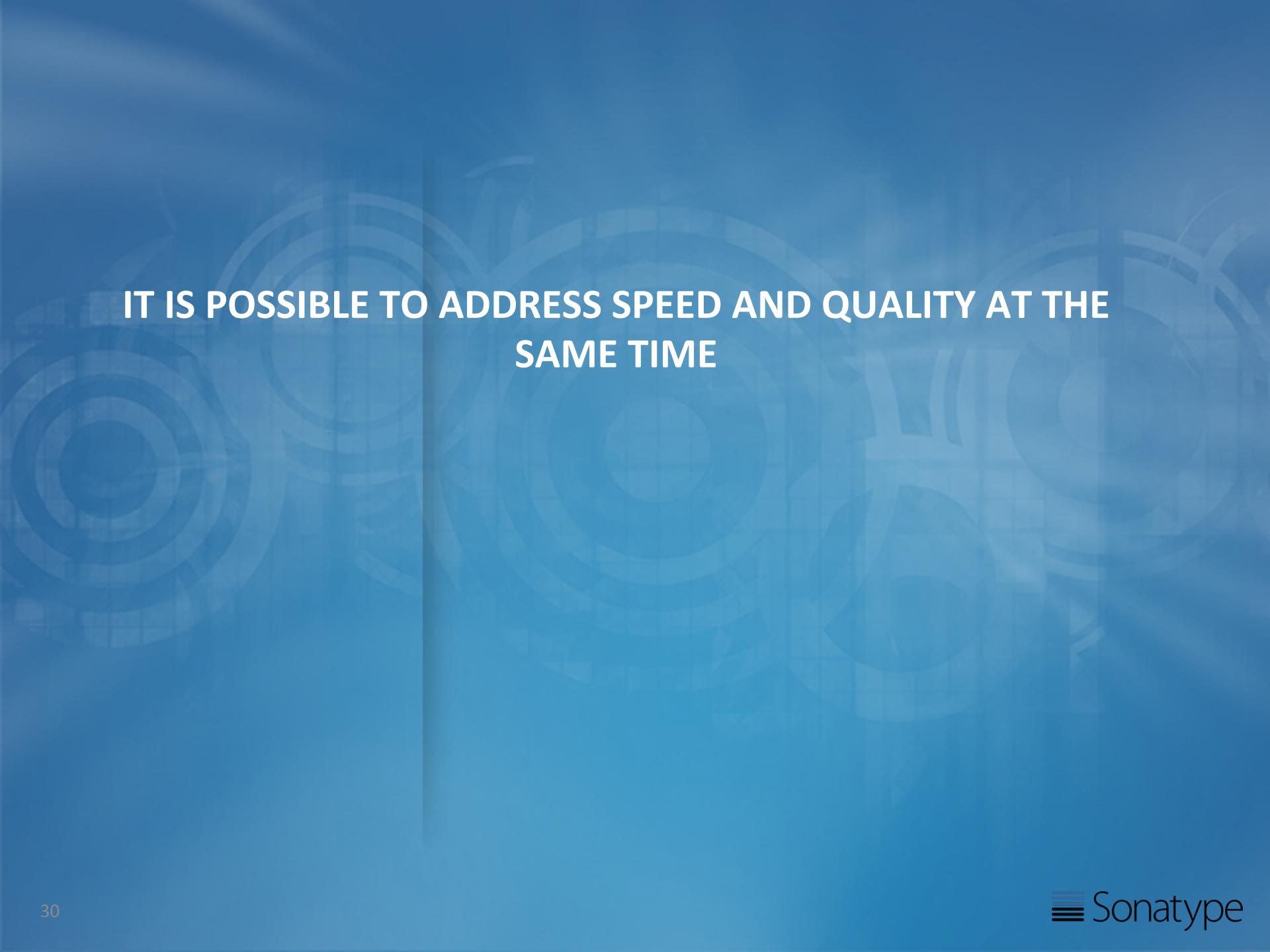
Cut the cord!

Point fingers!



RISK IN COMPONENTS

-
- The diagram features a large red circle on the left labeled "RISK IN COMPONENTS". To its right, a light gray circle contains four smaller red circles arranged in a vertical line. Each small red circle is connected by a red curved line to a text block describing a risk:
- Component usage has **exploded**
 - Applications are the primary vector of **attack**
 - There is a proliferation of **flawed** components
 - Current approaches **can't handle** the complexity at speed
- A faint watermark "Sonatype" is visible in the background.



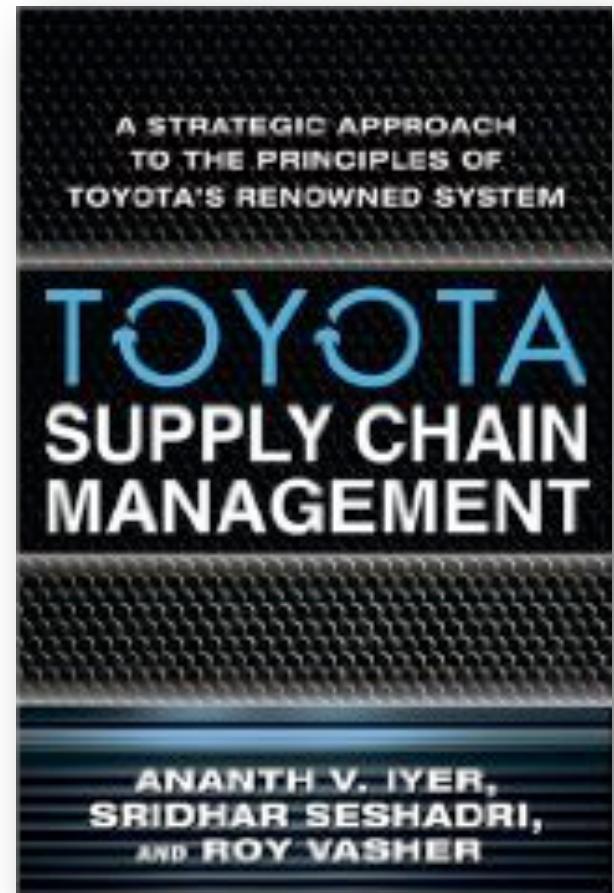
**IT IS POSSIBLE TO ADDRESS SPEED AND QUALITY AT THE
SAME TIME**



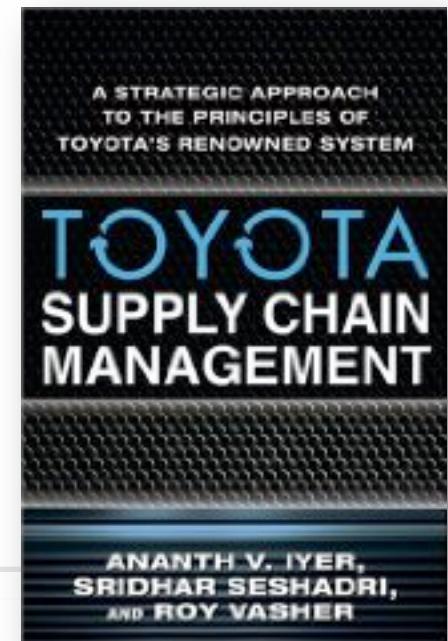
We are not the first **INDUSTRY** to
face this **CHALLENGE**

Comparing the Prius and Volt

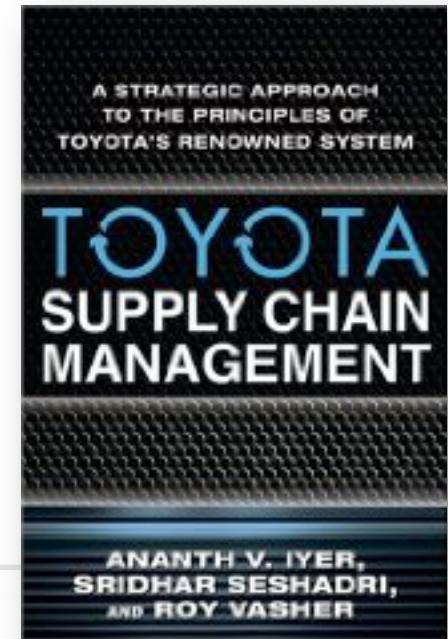
	Toyota Prius	Chevy Volt
Cost:	\$24,200	\$39,900
Units per month:	23,294	1,788
Plant Suppliers	125	800
Firm-Wide Suppliers	224	5,500
In-House Production	27%	54%



- Variety of products offered
- Velocity of product flow
- Variability of outcomes against forecast
- Visibility of processes to enable learning



- Variety of *software produced*
- Velocity of *software delivery*
- Variability of outcomes against forecast
- Visibility of processes to enable learning



Create Awareness

Empower

Govern

Monitor

Create Awareness

“Unless problems are seen, they will not be solved. Systems need to be in place to report ideas, problems, deviations, and potential issues with no delay.”

Fast so it can be
Continuous

Focus on the
attributes so it can
be automated

EASY TO CONSUME

Provide stakeholders actionable, easy to consume information to remediate problems

MyApp - 2014-05-20 - Build Report

Summary Policy Security Issues License Analysis

Scope of Analysis

27 COMPONENTS IDENTIFIED
98% OF ALL COMPONENTS ARE OPEN SOURCE

2 POLICY ALERTS AFFECTING 28 COMPONENTS
7 SECURITY ALERTS AFFECTING 7 COMPONENTS
8 LICENSE ALERTS

Dashboard Component Details

cobertura : cobertura :1.6

APPLICATION

- Import : LSS
 - License-Copyleft: 8
 - Architecture-Quality: 1
 - License-Observed Only: 1
- Import : Phoenix
 - License-Copyleft: 8
 - Architecture-Quality: 1
 - License-Observed Only: 1
- MyOrg : Sentinel
 - Unpopular: 3
 - Too Old: 3
- MyOrg : Epic
 - Copyleft: 5

Security Issues

How bad are the vulnerabilities and how many are there?

Critical (8-10)	Severe (4-7)	Moderate (1-3)	No Threat (0)
4	31	3	17

Threat Level: 0, 2, 4, 6, 8, 10, 12, 14, 16

The summary of security issues demonstrates the breakdown of vulnerabilities based on severity and the threat level it poses to your application.

Dependency Depth: 1, 2, 3, 4, 5

License Analysis

What type of licenses and how many of each?

Critical (8-10)	Severe (4-7)	Moderate (1-3)	No Threat (0)
2	2	4	17

8% 16% 8% 8%

The summary of license analysis demonstrates the number of licenses detected in each category.

Dependency Depth: 1, 2, 3, 4, 5

Filters ▾

Policy Summary

CATEGORY	COUNTS	DELTA	WEEKLY DELTAS	12 WEEK TREND
New	828	▲ 782	[Graph]	[Graph]
Fixed	121	▲ 100	[Graph]	[Graph]
Unresolved	770	▲ 682	[Graph]	[Graph]

Newest By Component By Application

	TOTAL RISK ▾	Critical	Severe	Moderate	Low
RELEASE *	801	259	259	150	133
RELEASE *	639	152	201	120	166
BUILD *	639	152	201	120	166
BUILD *	218	51	117	24	26
BUILD *	218	51	117	24	26
RELEASE *	218	51	117	24	26
	193	34	110	24	25
	185	27	35	123	0
BUILD *	155	18	35	102	0
STAGE RELEASE *	94	18	10	66	0
STAGE RELEASE *	60	45	15	0	0

Empower

“Unless someone is capable of solving a problem that might arise within the boundaries set for him or her, that person will be unable to contribute to the problem solving process.”

Since it's **fast** and
automated everyone
is on the same page

Precise

BE SPECIFIC

No Noise!



- There is a world of difference between saying "Struts is approved" and saying "Struts 2.3.16.1 is **good** and Struts 2.3.15.0 ANY OLDER VERSION will **get your system owned**"

Scan found 50,313 “issues” →



Real issue count: 204 →

Contextual Results

WHY CONTEXT MATTERS

- SQL Injection vulnerabilities don't affect applications without databases.
- CopyLeft may not be a problem for internal applications or services.
- I need information that applies to my application.



Actionable

PROVIDE A SOLUTION

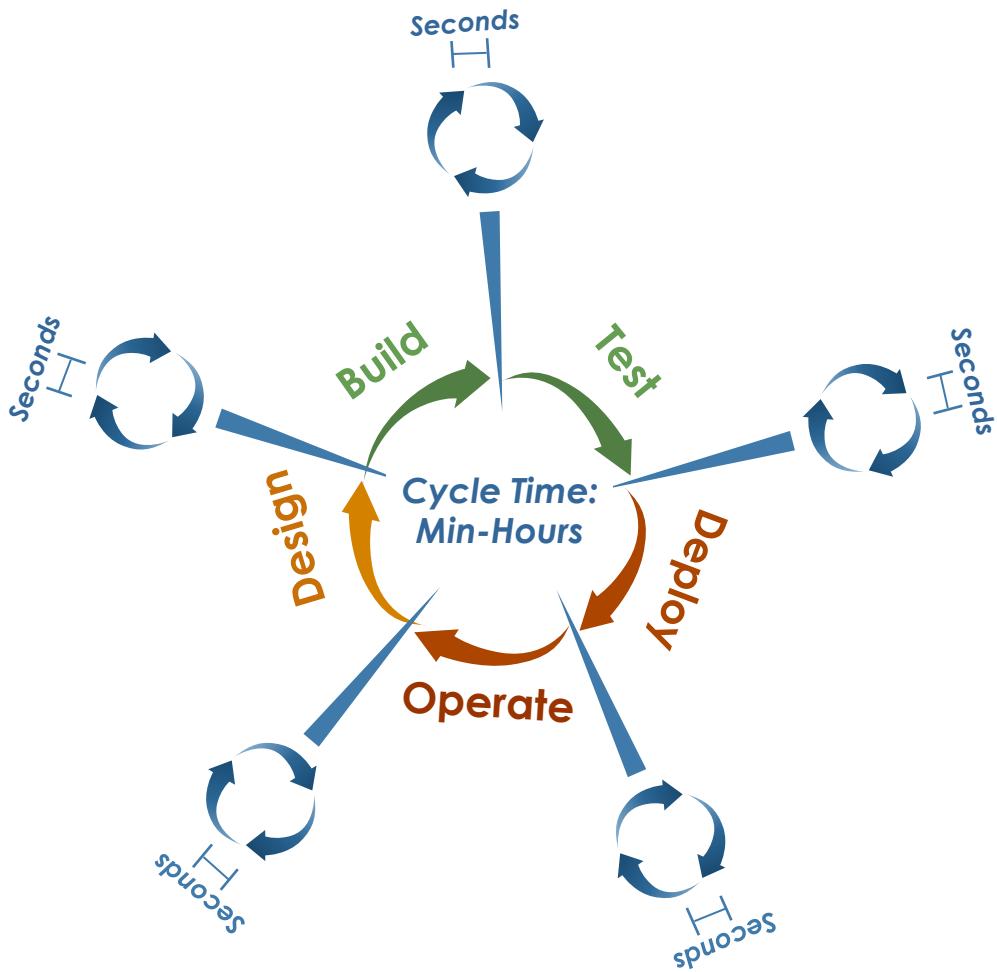
- Now that you've told me about a problem, tell me what I can do to fix it.
- Suggest alternatives.
- Even if I don't completely understand the risk,
if you show me an easy fix, I will take it.

Govern

“Actions have to be taken within a set of constraints, and they must conform to certain standards.”

Across the lifecycle

CYCLE TIME SYNERGY



Continuous Governance for Continuous Delivery

- No Interruption
- Entire Lifecycle
- Solve Early
- Avoid Rework

Go Fast **AND** Sleep at Night

The ability to govern
in an automated way
clears roadblocks

Monitor

“As experience with solving problems is obtained, greater awareness of other areas that might be affected needs to be created.”

Monitor process
improvements and
refine

Know where
components are
used to you can
respond



Applications don't age like wine,
**THEY ROT
LIKE MILK**



System.exit(0);



Questions?