THE NATIONAL NEED FOR SOFTWARE UNDERSTANDING

Douglas Ghormley
Senior Scientist
Sandia National Laboratories
March 20, 2024

For more information on this topic, email suns@sandia.gov.
Security News This Week: Russia's SolarWinds Hack Is a Historic Mess

All the most important stories about the biggest hack in years.

NORMALLY WE USE this space to round up the biggest stories from all reaches of the cybersecurity world. This week, we're making an exception, because there's really only one story: how Russia pulled off the biggest espionage hack on record.
Microsoft confirms it was also breached in recent SolarWinds supply chain hack

Microsoft denies that hackers pivoted to production systems and abused its software to attack customers.

By Catalin Cimpanu for Zero Day | December 17, 2020 -- 23:46 GMT | Topic: Security

The vast majority of these victims are US government agencies, such as:

- The US Treasury Department
- The US Department of Commerce’s National Telecommunications and Information Administration (NTIA)
- The Department of Health's National Institutes of Health (NIH)
- The Cybersecurity and Infrastructure Agency (CISA)
- The Department of Homeland Security (DHS)
- The US Department of State
- The National Nuclear Security Administration (NNSA) *(also disclosed today)*
- The US Department of Energy (DOE) *(also disclosed today)*
- Three US states *(also disclosed today)*
- City of Austin *(also disclosed today)*
According to the news, Microsoft confirmed that it was also breached in the recent SolarWinds hack. The vast majority of these victims were government agencies and businesses.

- The US Treasury Department
- The US Department of Commerce’s National Telecommunications and Information Administration (NTIA)
- The Department of Health’s National Institutes of Health (NIH)
- The Cybersecurity and Infrastructure Agency (CISA)
- The Department of Homeland Security (DHS)
- The US Department of State
- The National Nuclear Security Administration (NNSA) (also disclosed today)
- The US Department of Energy (DOE) (also disclosed today)
- Three US states (also disclosed today)
- City of Austin (also disclosed today)
According to the news, Microsoft confirmed SolarWinds was also breached in a recent SolarWinds security fiasco. The news also notes:

- The US Department of State
- The National Nuclear Security Administration (NNSA) (also disclosed today)
- The US Department of Energy (DOE) (also disclosed today)
- Three US states (also disclosed today)
- City of Austin (also disclosed today)
ACCORDING TO THE NEWS

Microsoft confirmed they also breached in Justice Department, federal court system hit by Russian hack

By ERIC TUCKER and FRANK BAJAK January 6, 2021

SolarWinds: The more we learn, the worse it looks

While you’ve been distracted by the holidays, coronavirus, and politics, the more we learn about the SolarWinds security fiasco, the worse it looks.

- The US Department of State
- The National Nuclear Security Administration (NNSA) (also disclosed today)
- The US Department of Energy (DOE) (also disclosed today)
- Three US states (also disclosed today)
- City of Austin (also disclosed today)
SolarWinds Hackers Breach Email Security Provider Mimecast, Compromise Customers’ Microsoft 365 Exchange Certificates

ALICIA HOPE · JANUARY 19, 2021

Suspected Russian hackers attributed to the worst supply chain attack breached email security provider Mimecast affecting a subset of its customers, the company said.

Mimecast said that Microsoft’s security experts notified the company of “a sophisticated threat actor” who hijacked its certificates used to connect to Mimecast customers’ Microsoft 365 Exchange products.

- Three US states (also disclosed today)
- City of Austin (also disclosed today)
If it were your job to detect such things in software \textit{before it is put into use}, what would you do?
WAYS OF DEALING WITH SOFTWARE RISK TODAY

Ask the Developer
Run Some Tests
Check the Signature
Investigate
WAYS OF DEALING WITH SOFTWARE RISK TODAY

- Ask the Developer
- Run Some Tests
- Check the Signature
- Investigate

Cummins Fined Record $1.6B for RAM Emissions Defeat Devices

© December 31, 2023  News Editor  © Latest News, RSS, Transport  © Comments Off
WAYS OF DEALING WITH SOFTWARE RISK TODAY

Ask the Developer  Run Some Tests  Check the Signature  Investigate

Contractor admits planting logic bombs in his software to ensure he’d get new work

Cummins Fined Record $1.6B for Defeat Devices

Logic bombs created periodic malfunctions that only he knew how to fix.
WAYS OF DEALING WITH SOFTWARE RISK TODAY

- Ask the Developer
- Run Some Tests
- Check the Signature
- Investigate

Contractor admits planting backdoor in Israeli military software

Code kept Secret for Years Reveals Its Flaw—a Backdoor

A secret encryption cipher baked into radio systems used by critical infrastructure workers, police, and others around the world is finally seeing sunlight. Researchers say it isn’t pretty.
WAYS OF DEALING WITH SOFTWARE RISK TODAY

Evasive Malware Detects and Defeats Virtual Machine Analysis

Advanced malware solutions ("sandboxes") traditionally use virtual machines (VM) to analyze suspicious objects to find out if they are malicious. However, advanced malware is capable of detecting the presence of the virtual machine technology used by conventional sandboxes and leveraging this weakness to evade detection.
WAYS OF DEALING WITH SOFTWARE RISK TODAY

- Ask the Developer
- Run Some Tests
- Check the Signature
- Investigate

Advanced malware solutions ("sandboxes") traditionally use virtual machines (VM) to analyze suspicious objects to find out if they are malicious. However, advanced malware is capable of detecting the presence of the virtual machine technology used by conventional sandboxes and leveraging this weakness to evade detection.

Software Testing is Necessary But Not Sufficient for Software Trustworthiness
WAYS OF DEALING WITH SOFTWARE RISK TODAY

- Ask the developer
- Run static tests
- Check the signature
- Investigate

Advanced malware solutions ("sandboxes") traditionally use virtual machines (VM) to analyze suspicious objects to find out if they are malicious. However, advanced malware is capable of detecting the presence of the virtual machine technology used by conventional sandboxes and leveraging this weakness to evade detection.


...it is important to realize that testing alone is rarely sufficient to establish high levels of dependability.
In December, an unauthorized user accessed GitHub’s systems and stole three encrypted code-signing certificates: one Apple-issued Developer ID certificate and two DigiCert-issued code-signing certificates for its desktop and Atom applications.

Another security breach at Micro-Star International (MSI) resulted in a software supply chain attack, where hackers had access to private signing keys for MSI’s firmware and Intel’s UEFI.
In December, an unauthorized user accessed GitHub's systems and stole three encrypted code-signing certificates: one Apple-issued Developer ID certificate and two DigiCert-issued code-signing certificates for its desktop and Atom applications.

Another security breach at Micro-Star International (MSI) resulted in a software supply chain attack, where hackers had access to private signing keys for MSI's firmware and Intel's UEFI.

WAYS OF DEALING WITH SOFTWARE RISK TODAY

1. Ask the Developer
2. Run Some Tests
3. Check the Signature
4. Investigate

Hackers are selling legitimate code-signing certificates to evade malware detection

Code-signed apps are harder to detect by network security appliances, making it easier to sneak malware onto a vulnerable system. The downside? Certificates aren't cheap — and hackers usually are.

Dave Roche
August 1, 2023

By Zack Whittaker for Zero Day | February 22, 2015 - 13:00

GMT (00.00 PST) | Topic: Security
In December, an unauthorized user accessed GitHub’s systems and stole three encrypted code-signing certificates: one Apple-issued Developer ID certificate and two DigiCert-issued code-signing certificates for its desktop and Atom applications.

Another security breach at Micro-Star International (MSI) resulted in a software supply chain attack, where hackers had access to private signing keys for MSI’s firmware and Intel’s UEFI.

WAYS OF DEALING WITH SOFTWARE RISK TODAY

1. Ask the developer
2. Run some tests
3. Check the signature
4. Investigate

The misuse of code signing certificates is so widespread that a larger percentage of malware downloaded to computers is digitally signed than that of benign software programs.
WAYS OF DEALING WITH SOFTWARE RISK TODAY

1. Ask the developer
2. Run static tests
3. Check the signature
4. Investigate

Tens of millions of biz Dell PCs smacked by privilege-escalation bug in bundled troubleshooting tool
If you don’t have auto-update switched on, time to patch

Laurie Clarke  Tue 11 Feb 2020 // 15:01 UTC
WAYS OF DEALING WITH SOFTWARE RISK TODAY

- Ask the developer
- Run security tests
- Check the signature
- Investigate

Between Jan. 20 and March 11, researchers observed APT41 exploiting vulnerabilities in Citrix NetScaler/ADC, Cisco routers and Zoho ManageEngine Desktop Central as part of the widespread espionage campaign.
WAYS OF DEALING WITH SOFTWARE RISK TODAY

Between Jan. 20 and March 11, researchers observed APT41 exploiting vulnerabilities in Citrix NetScaler/ADC, Cisco routers and Zoho ManageEngine Desktop Central as part of the widespread espionage campaign.

Apple mistakenly approved a widely used malware to run on Macs
THE SOFTWARE SUPPLY CHAIN IS GLOBAL

Source: An advertising video from the Czech Republic encouraging the outsourcing of code development
In a global economy it is infeasible to only use software developed in “trustworthy” countries.

Source: An advertising video from the Czech Republic encouraging the outsourcing of code development
SUPPLY CHAIN ASSURANCE – A MAJOR GAP

Company ABC

Company DEF

Company XYZ

The Board

✓

✓

✓

✓

✓

✓

✓

7f 41 2a 0 33 00 00 0 ff 04 c7 c3 c9 cd 7c 01 00 f 3d aa 82 0e f7 8c 93 66 68 00 00 bb 66 83 77 cd c3 b3 9c cc 3a 7e 7f
SUPPLY CHAIN ASSURANCE – A MAJOR GAP

SUNSPOT: An Implant in the Build Process

January 11, 2021  Crowdstrike Intelligence Team  Research & Threat Intel

- SUNSPOT monitors running processes for those involved in compilation of the Orion product and replaces one of the source files to include the SUNBURST backdoor code.
- Several safeguards were added to SUNSPOT to avoid the Orion builds from failing, potentially alerting developers to the adversary's presence.

Orion Source Code Replacement

When SUNSPOT finds the Orion solution file path in a running MsBuild.exe process, it replaces a source code file in the solution directory, with a malicious variant to inject SUNBURST while Orion is being built.

While SUNSPOT supports replacing multiple files, the identified copy only replaces InventoryManager.cs.
SUPPLY CHAIN ASSURANCE – A MAJOR GAP

The Board

Company ABC

✓

Company DEF

✓

Company XYZ

✓
Russia's SolarWinds hack has no easy fix, cybersecurity company says

Efforts to assess the impact of a more than seven-month-old cyberespionage campaign blamed on Russia – and boot the intruders – remain in their early stages.

By The Associated Press

Carmakal said he believed software companies were prime targets because hackers of this caliber will seek to use their products — as they did with SolarWinds' Orion module — as conduits for similar so-called supply-chain hacks.

WHAT’S NEEDED MORE THAN ALL OF THESE?

Ask the Developer
Run Some Tests
Check the Signature
Investigate

Analyze the Software’s Behavior!
Assess the potential behavior of the software itself to answer key national security & critical infrastructure mission questions.
WHAT’S NEEDED MORE THAN ALL OF THESE?

Ask the Developer  Run Some Tests  Check the Signature  Investigate

All of the questions above are **proxies** for the one thing that matters most:

*What can the software actually do?*

Confidently answering questions about existing software requires technical analysis of the potential behaviors of the software.
The USG has integrated 3rd-party software into every facet of national security (NS), critical infrastructure (CI), and government. This software regularly exhibits undesirable behaviors that put mission at risk.

To ensure NS and CI mission success, we must pose and answer a variety of mission-specific questions about software's potential behavior.
“MISSION QUESTIONS” ABOUT SOFTWARE IN A DESTROYER

- Could sensitive communications be relayed off-ship?
- Will the radar software lie under certain conditions?
- Could false commands be issued by this software?
- Can my secure comms software send unencrypted messages?
- Can the software leak my ship’s location?
- Is there hidden, triggered behavior in our weapon’s targeting systems?
"MISSION QUESTIONS" ABOUT SOFTWARE ACROSS MISSIONS

- Could our sensitive data be changed without authorization?
- Could our sensitive data be leaked?
- Is this software vulnerable to attack XYZ?
- Does this software have hidden ransomware?
- What indicators does this malware have?
- Could our sensitive data be changed without authorization?
THE FULL SCOPE OF THE PROBLEM

Examples are entirely notional, for illustration purposes only.
THE FULL SCOPE OF THE PROBLEM

Examples are entirely notional, for illustration purposes only.
Examples are entirely notional, for illustration purposes only.

Ideally, mission owners should be able to routinely analyze any mission critical software to answer any mission question.
THE FULL SCOPE OF THE PROBLEM

Ideally, mission owners should be able to routinely analyze any mission critical software to answer any mission question.

- Could our sensitive data be changed without authorization?
- Does this software have hidden ransomware?
- Is there a kill switch hidden in my propulsion systems?
- What indicators does this malware have?
- Is there hidden, triggered behavior in our weapon’s targeting systems?
- Could sensitive communications be relayed off-ship?
- Can my ship’s location be tracked?
- Could false commands be issued by this software?
- Can my secure comms software send unencrypted messages?
- Could our sensitive data be leaked?
- Does this software have a backdoor?
- Could our sensitive data be changed without authorization?
- Does this software have hidden ransomware?
- Is there a kill switch hidden in my propulsion systems?
- What indicators does this malware have?

Examples are entirely notional, for illustration purposes only.
Today, an agency needing to analyze one piece of software to answer a mission question can fund an effort to do that analysis.

Is there hidden, triggered behavior in our weapon’s targeting systems?

Could our sensitive data be changed without authorization?

Does this software have hidden ransomware?

Could sensitive communications be relayed off-ship?

Can my ship’s location be tracked?

Does this software have a backdoor?

Could false commands be issued by this software?

Could our secure communications software send unencrypted messages?

Could our sensitive data be leaked?

Does this software have hidden malware?

What indicators does this malware have?

Is there a kill switch hidden in my propulsion systems?

Is this software vulnerable to attack XYZ?

Examples are entirely notional, for illustration purposes only.
HOW WE OPERATE TODAY

Today, an agency needing to analyze one piece of software to answer a mission question can fund an effort to do that analysis.

- Could our sensitive data be changed without authorization?
- Does this software have hidden ransomware?
- Is there hidden, triggered behavior in our weapon’s targeting systems?
- Could sensitive communications be relayed off-ship?
- Can my ship’s location be tracked?
- Does this software have a backdoor?
- Is there a kill switch hidden in my propulsion systems?
- What indicators does this malware have?
- Could false commands be issued by this software?
- Could our sensitive data be leaked?
- Does this software have hidden ransomware?
- Could sensitive communications be relayed off-ship?
- Can my ship’s location be tracked?
- Does this software have a backdoor?
- Is there a kill switch hidden in my propulsion systems?
- What indicators does this malware have?
- Could false commands be issued by this software?
- Could our sensitive data be leaked?

This is a major undertaking.

Examples are entirely notional, for illustration purposes only.
CHARACTERISTICS OF MODERN SOFTWARE SYSTEMS

- **Dynamical**, not Static
- **Nonlinear**, not Linear
- **Deterministic**, not Probabilistic
- **Discrete**, not Continuous
- **Universal**, not Specific
- **Complex**, not Simple

This combination of characteristics is the hardest of the set of options for analysis.

Also, these same characteristics are what makes software so effective in meeting functional requirements.
CHARACTERISTICS OF MODERN SOFTWARE SYSTEMS

- Dynamical
- Nonlinear
- Deterministic
- Discrete
- Universal
- Complex

Our lives are steeped in continuous systems.
CHARACTERISTICS OF MODERN SOFTWARE SYSTEMS

- Dynamical
- Nonlinear
- Deterministic
- Discrete
- Universal
- Complex

Our lives are steeped in continuous systems.
Our lives are steeped in continuous systems.

If x pounds is safe and y pounds is safe... then anything between [x, y] must also be safe.
Software is discrete, not continuous. Successfully testing software with inputs 2 and 4 tells you *nothing* about the behavior on input 3.

If x pounds is safe and y pounds is safe, then anything between $[x, y]$ must also be safe.
CHARACTERISTICS OF MODERN SOFTWARE SYSTEMS

- Dynamical
- Nonlinear
- Deterministic
- Discrete
- Universal
- Complex

We simulate simplicity with cheap complexity.
CHARACTERISTICS OF MODERN SOFTWARE SYSTEMS

- Dynamical
- Nonlinear
- Deterministic
- Discrete
- Universal
- Complex

We simulate simplicity with cheap complexity.

Car door locking systems should be simple.
Simple but custom is expensive.

A Door Control Unit (DCU) has a full CPU, runs a real-time OS.

$\$\$\$

< $2
THE SOFTWARE UNDERSTANDING GAP

The software understanding gap is expanding exponentially. The more it expands, the more it will impact national security and critical infrastructure missions.

When the USG switched from GOTS to COTS, the ability to analyze software was adequate, but that quickly changed.

This gap will continue to grow exponentially until software understanding solutions are developed.

1986: Fred Brooks “No Silver Bullet”
1994: Perry Memo
1996: Phrack Article, Smashing the Stack for Fun and Profit”
1997-Present: Software complexity begins to expand faster than the ability to analyze it
INHERENT RISK VS. RESIDUAL RISK

Residual risk is the risk that remains after inherent risk has been partially mitigated.

Inherent Risk
The risk to an entity in the absence of any direct or focused actions to alter its severity.

Risk Mitigated by Security Controls
Security controls are applied to mitigate inherent risk.

Residual Risk
Portion of risk remaining after security controls have been applied.

Understanding inherent risk is the first step in the risk assessment process.

For software, the lack of adequate software understanding capability means that risk assessors cannot effectively implement the first step in the process, rendering the rest of the process fundamentally flawed.
Today, an agency needing to analyze one piece of software to answer a mission question can fund an effort to do that analysis.

Could our sensitive data be changed without authorization?
Could sensitive communications be relayed off-ship?
Can my ship's location be tracked?
Could our sensitive data be leaked?

Does this software have hidden ransomware?
Does this software have a backdoor?

What indicators does this malware have?
Could false commands be issued by this software?

Is there hidden, triggered behavior in our weapon's targeting systems?

Is there a kill switch hidden in my propulsion systems?

This is a major undertaking.

Examples are entirely notional, for illustration purposes only.
Today, an agency needing to analyze one piece of software to answer a mission question can fund an effort to do that analysis.

Examples are entirely notional, for illustration purposes only.
Today, an agency needing to analyze one piece of software to answer a mission question can fund an effort to do that analysis.

Examples are entirely notional, for illustration purposes only.
Today, an agency needing to analyze one piece of software to answer a mission question can fund an effort to do that analysis.

**Examples are entirely notional, for illustration purposes only.**
Today, an agency needing to analyze one piece of software to answer a mission question can fund an effort to do that analysis.

This is, in effect, the current approach. The entire GDP of the nation is insufficient to meet the need with this approach.

We have adjusted our policy, planning, procedures, expectations, etc. to fit the lack of capability.

Today, we put software into use without knowing the answers to questions like these. We discover mission-threatening behavior after the software is placed into service.

Consequently, the nation is currently facing unmeasurable, unbounded risk from software.
Today, an agency needing to analyze one piece of software to answer a mission question can fund an effort to do that analysis.

**HOW WE OPERATE TODAY**

Can my ship's location be tracked?

Could false commands be issued by this software?

Is there hidden, triggered behavior in our weapon's targeting systems?

Can my secure comms software send unencrypted messages?

Could sensitive communications be relayed off-ship?

Could our sensitive data be changed without authorization?

Could our sensitive data be leaked?

Does this software have hidden ransomware?

Does this software have a backdoor?

What indicators does this malware have?

Is this software vulnerable to attack XYZ?

Can we envision a future where this problem is tractable?

What is holding us back from getting there?

Examples are entirely notional, for illustration purposes only.
5 USG representatives co-convened ~30 technical SMEs from 10 USG research groups to discuss the need for a national capability for software understanding.

QUESTIONS PRESENTED

1. What are technical impediments to a national software understanding capability?
2. How can the USG support the expansion of software understanding?
3. What are near-term R&D funding priorities?

Top 5 Issues:

1. Lack of Unified Vision
2. Community Building
3. Lack of Sharing
4. Funding
5. Metrics and Benchmarks

Many expressed the opinion that a national cyber crisis is inevitable unless we revolutionize the way we analyze software.
### SUNS 2023 WORKSHOP

#### Co-Conveners

<table>
<thead>
<tr>
<th>CISA</th>
<th>NSA</th>
<th>NNSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Senior Technical Director for Cybersecurity Division</td>
<td>Technical Director of Research</td>
<td>Director, Nuclear Enterprise Assurance Division</td>
</tr>
<tr>
<td>Associate Chief of Strategic Technology</td>
<td>Technical Director of Cybersecurity</td>
<td></td>
</tr>
</tbody>
</table>

#### Open Sessions

**53 Attendees from:**
- Army/ESIC
- CISA
- DARPA
- DHS/S&T
- DIA
- MIT-LL
- NIST
- NSA
- ODNI
- OUSD R&E
- PNNL
- SEI
- SNL
- ZRA

#### Closed Sessions

**29 Technical SMEs from:**
- DARPA
- CISA
- IDA/CCS
- GTRI
- LLNL
- MIT-LL
- NSA
- PNNL
- SEI
- SNL
10x-100x+ improvement in software understanding capabilities is possible, but progress is currently prevented by lack of a centralized vision, funding that is 10x+ too low, inability to collaborate, and other non-technical issues.

80% of SMEs expressed opinions consistent with this statement.

The scope of a national software understanding vision should include foundational research.

100% of SMEs expressed opinions consistent with this statement.
OUR CONCLUSIONS BASED ON THE SUNS 2023 DISCUSSIONS

#1 Radically improved technical capabilities for software understanding are possible.

#2 A unified national effort to revolutionize our software understanding capabilities is necessary to meet current and future mission needs. We are far from being on track today.

#3 The nation that learns to best analyze and reason about software artifacts will dominate global geopolitics for the next century.
AN OBSERVATION ABOUT SOFTWARE ANALYSIS TOOLS

analyzer \approx f(\text{mission\_question}, \text{program\_under\_test}, \text{resource\_tradeoffs})
AN OBSERVATION ABOUT SOFTWARE ANALYSIS TOOLS

analyzer \approx f(mission\_question, program\_under\_test, resource\_tradeoffs)

This term explains why a tool designed for one purpose is ill suited for others.

E.g., a tool designed to detect authentication backdoors is ill suited to identifying information leaks.
AN OBSERVATION ABOUT SOFTWARE ANALYSIS TOOLS

\[ \text{analyzer} \approx f(\text{mission_question}, \text{program_under_test}, \text{resource_tradeoffs}) \]

This term explains why tools don’t work well across many programs.

E.g., what’s needed to analyze real-time flight controllers and web servers are different.
AN OBSERVATION ABOUT SOFTWARE ANALYSIS TOOLS

\[ \text{analyzer} \approx f(\text{mission\_question}, \text{program\_under\_test}, \text{resource\_tradeoffs}) \]

Varying resource and accuracy tradeoffs across mission applications is captured by this term.

E.g., national security missions may choose to invest far more computational resources than a typical laptop user.
When we build analysis tools without coordination, we need a different tool for each combination of question, program, and resource tradeoffs:

$$|\text{analyzer}| = |\text{mission questions}| \times |\text{programs}| \times |\text{tradeoffs}|$$
AN OBSERVATION ABOUT SOFTWARE ANALYSIS TOOLS

analyzer ≈ f(mission_question, program_under_test, resource_tradeoffs)

This is, in effect, the current approach. The entire GDP of the nation is insufficient to meet mission needs in software understanding using this approach.

|analyzer| = |mission questions| * |programs| * |tradeoffs|
We need automated software understanding tools designed to scale across varying:

- Mission Questions
- Program executables
- Resource tradeoffs
To build an analysis tool to help answer a given question for a given program, we need to model the execution of the program in order to analyze its potential behavior.

Examples are entirely notional, for illustration purposes only.
To build an analysis tool to help answer a given question for a given program, we need to model the execution of the program in order to analyze its potential behavior.
CONSIDER A TOOL FOR A SINGLE ANALYSIS TASK

Most tools today are largely monolithic, hard-coding various modeling decisions.

Examples are entirely notional, for illustration purposes only.
Observations about monolithic analysis tools

Observation #1

- Poor models mean poor analysis results
- There is no single “right” answer for:
  - All mission questions
  - All programs
  - All customer needs, resources, and risk appetite

Monolithic, bespoke implementations are not generally reusable.

Observation #2

- Each analysis tool is a “chain link problem”
- It’s only as good as its weakest link

Foundational, principled approaches for reusable components are disincentivized.
SCALABILITY DEMANDS DIFFERENT APPROACHES

Examples are entirely notional, for illustration purposes only.
SCALABILITY DEMANDS DIFFERENT APPROACHES

The current USG R&D landscape is pushing the community the wrong way.
SCALABILITY DEMANDS DIFFERENT APPROACHES

Technical Challenges

Cross-Agency Span

This challenge spans government.

Solution approaches could span agencies with a coordinated effort.

No department or agency has the charter or ability to explore solutions that scale to the entire USG.
**Recommendation #1: Make a national decision to address the software understanding gap**

- The White House must direct coordination across Departments and Agencies
  - Whole of government & society effort
  - Senior technical SMEs directing investments
  - Continuity of effort across the R&D community

**Recommendation #2: Create a cross-agency Software Understanding for National Security Oversight Council (SUNSOC)**

- Provide national coordination across community engagements
- Establish collaborative agreements and environments
- Provide technical SME direction by developing and maintain a national R&D roadmap
- Manage national investments in software understanding
- Cultivate the software understanding community as a national resource
QUESTIONS?