Attack and Defense in the Cyber-Physical World

Ning Zhang

https://cybersecurity.seas.wustl.edu
Why should you care? It doesn’t speed up your PC for sure!

57% of IT decision makers think discussions around AI in cyber security are just hype.
Tremendous societal impacts of cybersecurity around the globe

"WannaCry" ransomware attack losses could reach $4 billion

Hackers remotely kill a jeep on the highway—with me in it

Russian Hacking and Influence in the U.S. Election
Complete coverage of Russia's campaign to disrupt the 2016 presidential election.

https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/
There is a demand for people in this field.
What is cyber security to you?

What do you think we do?
Security is a method of analyzing system

*Is this piece of code secure?*

```c
bool authenticate(const char * password)
{
    if(strncmp(password, “hello”, strlen(“hello”))==0)
        return true;
    else
        return false;
}
```
Security and Privacy
(my best effort categorization)

Security
- System Security
  - Architecture Security (sw/hw codesign, design tradeoffs)
  - Trustworthy Computing
  - Software Security (OS, App, Mobile)
  - Authentication (Oauth, Biometrics)
  - Cryptography
    - Core Crypto Theory
    - Applied Crypto
  - Adversarial AI
  - Human factor security
  - Communication Security
  - Network Protocol
    - Blockchain Decentralization

Privacy
- Censorship
- Confidential Computation
- Anonymous Communication
- Data Anonymization / Publishing
- Location Privacy
Computer Security 101

Achieve some *goal* against some *adversary*

System Goal / Security Service / Policy
Threat models
Mechanism

Common goals include:
Confidentiality – nobody can see my stuff except myself
Integrity – nobody can malicious modify my stuff
Availability – I should have access to my things when I need to
Today’s discussion

Three of my on-going research

- Software Security
- Confidential Computing
- Architecture Security
Software Security
Cyber Security
Capture the flag
How do we automatically find vulnerabilities?
Program Analyzers

<table>
<thead>
<tr>
<th>Report</th>
<th>Type</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>mem leak</td>
<td>324</td>
</tr>
<tr>
<td>2</td>
<td>buffer overflow</td>
<td>4,353,245</td>
</tr>
<tr>
<td>3</td>
<td>sql injection</td>
<td>23,212</td>
</tr>
<tr>
<td>4</td>
<td>stack overflow</td>
<td>86,923</td>
</tr>
<tr>
<td>5</td>
<td>dang ptr</td>
<td>8,491</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10,502</td>
<td>info leak</td>
<td>10,921</td>
</tr>
</tbody>
</table>

Slide credit: John Mitchell
Two options

Static analysis
- Automated methods to find errors or check their absence
  - Consider all possible inputs (in summary form)
  - Find bugs and vulnerabilities
  - Can prove absence of bugs, in some cases

Dynamic analysis
- Run instrumented code to find problems
  - Need to choose sample test input
  - Can find vulnerabilities but cannot prove their absence
Dynamic testing examines subset of behaviors.

Slide credit: John Mitchell
Static testing uses abstraction to consider all behaviors.
Tainting checkers

Tainted data accepted from source

Unvetted data taints other data transitively

Tainted data is used in an operator or function

Example Sinks: system() | printf() | malloc() | strcpy() | Sent to RDBMS | Included in HTML

Resultant Vulnerability: command injection | format string manip. | integer/buffer overflow | buffer overflow | SQL injection | cross site scripting

Slide credit: John Mitchell
<table>
<thead>
<tr>
<th>process timing</th>
<th>overall results</th>
</tr>
</thead>
<tbody>
<tr>
<td>run time</td>
<td>cycles done</td>
</tr>
<tr>
<td>0 days, 0 hrs, 8 min, 24 sec</td>
<td>0</td>
</tr>
<tr>
<td>last new path</td>
<td>total paths</td>
</tr>
<tr>
<td>0 days, 0 hrs, 1 min, 59 sec</td>
<td>812</td>
</tr>
<tr>
<td>last uniq crash</td>
<td>uniq crashes</td>
</tr>
<tr>
<td>0 days, 0 hrs, 3 min, 17 sec</td>
<td>8</td>
</tr>
<tr>
<td>last uniq hang</td>
<td>uniq hangs</td>
</tr>
<tr>
<td>0 days, 0 hrs, 3 min, 23 sec</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cycle progress</th>
<th>map coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>now processing</td>
<td>map density</td>
</tr>
<tr>
<td>0 (0.00%)</td>
<td>3158 (4.82%)</td>
</tr>
<tr>
<td>paths timed out</td>
<td>count coverage</td>
</tr>
<tr>
<td>0 (0.00%)</td>
<td>2.56 bits/tuple</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stage progress</th>
<th>findings in depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>now trying</td>
<td>favored paths</td>
</tr>
<tr>
<td>arith 8/8</td>
<td>1 (0.12%)</td>
</tr>
<tr>
<td>stage execs</td>
<td>new edges on</td>
</tr>
<tr>
<td>295k/326k (90.31%)</td>
<td>318 (39.16%)</td>
</tr>
<tr>
<td>total execs</td>
<td>total crashes</td>
</tr>
<tr>
<td>552k</td>
<td>63 (8 unique)</td>
</tr>
<tr>
<td>exec speed</td>
<td>total hangs</td>
</tr>
<tr>
<td>1114/sec</td>
<td>191 (10 unique)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>fuzzing strategy yields</th>
<th>path geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit flips</td>
<td>levels</td>
</tr>
<tr>
<td>447/75.5k, 59/75.5k, 59/75.5k</td>
<td>2</td>
</tr>
<tr>
<td>byte flips</td>
<td>pending</td>
</tr>
<tr>
<td>7/9436, 0/5858, 6/5950</td>
<td>812</td>
</tr>
<tr>
<td>arithmetics</td>
<td>pend fav</td>
</tr>
<tr>
<td>0/0, 0/0, 0/0</td>
<td>1</td>
</tr>
<tr>
<td>known ints</td>
<td>own finds</td>
</tr>
<tr>
<td>0/0, 0/0, 0/0</td>
<td>811</td>
</tr>
<tr>
<td>dictionary</td>
<td>imported</td>
</tr>
<tr>
<td>0/0, 0/0, 0/0</td>
<td>n/a</td>
</tr>
<tr>
<td>havoc</td>
<td>variable</td>
</tr>
<tr>
<td>0/0, 0/0</td>
<td>0</td>
</tr>
<tr>
<td>trim</td>
<td></td>
</tr>
<tr>
<td>0.00%/1166, 38.39%</td>
<td></td>
</tr>
</tbody>
</table>

[cpu: 15%]
Bugs to Detect

- Crash Causing Defects
- Null pointer dereference
- Use after free
- Double free
- Array indexing errors
- Mismatched array new/delete
- Potential stack overrun
- Potential heap overrun
- Return pointers to local variables
- Logically inconsistent code
- Uninitialized variables
- Invalid use of negative values
- Passing large parameters by value
- Underallocations of dynamic data
- Memory leaks
- File handle leaks
- Network resource leaks
- Unused values
- Unhandled return codes
- Use of invalid iterators

Slide credit: Andy Chou
Confidential Computing
Data Privacy
Internet of Things with Intelligence
Data Privacy

Who can access my data?
Access Control (System or Cryptographically)

How can they use my data?
User Agreement
Medical data privacy
Widespread availability of data
We are at the dawn of the age of data (Misused?)

Facebook Is Telling People Their Data Was Misused by Cambridge Analytica and They’re Furious

Facebook–Cambridge Analytica data scandal

The Facebook–Cambridge Analytica data scandal involves the collection of personally identifiable information of 87 million Facebook users[1] and reportedly a much greater number more[2] that Cambridge Analytica began collecting in 2014. The data was allegedly used to attempt to influence voter opinion on behalf of politicians who hired them. Following the discovery, Facebook apologized amid public outcry and risen stock prices. The way that Cambridge Analytica collected the data was called "inappropriate".[3]
**An ideal world for data privacy**

**Confidentiality Protection of User Data**
- Data shall always be encrypted with user controlled keys

**Verifiable user-controlled fine-grained utilization of data**
- **Access control:** User can add terms and conditions as *who* can access *what data* for *which purpose* during *what time* under *what condition*
- **Usage enforcement:** Data can only be used for approved purposes

**Non-repudiable recording of data use**
- When data is used, there will be *irreversible proof* of *how* it was used by *whom*
Trustworthy Non-repudiable recording?
What is blockchain?
We are done ?
Unfortunately, it is never that easy!
PrivacyGuard Framework

**Data Plane**
- Cloud
- Service Support
- Network
- Devices

- Encrypted Storage
- Trustworthy Data Utilization
- iData Agent

**Data Control Plane**
- Data Consumers
- Data Owners

- Block Chain Smart Contract
- Trustworthy Data Usage Tracking

- Remote Attestation
- Protected Data Provenance
- Data Usage Smart Contract
Architecture Security
ARM TrustZone – resources are divided into two worlds

More than processor extension, TrustZone is system-wide security extension.

NonSecure(NS) bit is added to resource and bus
Attack overview – prime and probe
Two challenges

Prime - Triggering the cache contention
– Allocation of attack memory that will map the same cache sets of the victim process inside TrustZone.
• Lack of virtual-to-physical address mapping

Probe - Detection of the cache contention
– Detect changes in the cache state as a result of the resource contention.
• Lack of high accuracy timer
• Lack of ability for fine grained cache manipulation
The magic of cache access to leakage of cryptographic keys

To speed up GF operations, precomputed tables are used instead, $T_0$ to $T_{4r}$, 4KB size

\[ C[j] = T_i[X_{i_{\text{max}}}] \otimes K_{i_{\text{max}}}[j] \]

\[
X_{i+1} = \begin{cases} 
X_i \otimes K_i & i = 0 \\
MC(SR(SB(X_i))) \otimes K_i & 0 < i < i_{\text{max}} \\
SR(SB(X_i)) \otimes K_i & i_{\text{max}} 
\end{cases}
\]
Key extraction from the normal world OS

Completes key extraction in 2.5 seconds
Key extraction from the normal world unprivileged app

Completes key extraction in 15 mins
Broad Direction

- Software Security
- Trusted Computing
- CPS Security
New Attacks that interleave the cyber and physical world
What can you expect when you get out of my lab?
But what your life would be like while in my lab - problems
You will ...
What your life would be like while in my lab – Solution space

<table>
<thead>
<tr>
<th>Sys Sec</th>
<th>Sw Sec</th>
<th>CPS Sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Applied cryptography</td>
<td>• Program analysis</td>
<td>• Real-time system</td>
</tr>
<tr>
<td>• Operating system</td>
<td>• Static analysis</td>
<td>• Embedded System</td>
</tr>
<tr>
<td>• Assembly language</td>
<td>• Dynamic analysis</td>
<td>• Analog system</td>
</tr>
<tr>
<td>• Computer organization and</td>
<td>• Virtualization</td>
<td>• Digital Communication</td>
</tr>
<tr>
<td>architecture</td>
<td>• Fuzzing</td>
<td>• Wireless Communication</td>
</tr>
<tr>
<td></td>
<td>• Symbolic execution</td>
<td>• Physics</td>
</tr>
<tr>
<td></td>
<td>• Compiler module</td>
<td>• Chemistry</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Biology</td>
</tr>
</tbody>
</table>