Industrial control systems malware and integrity

Results from the Preemptive research project
Critical Infrastructures (CI) and Industrial Control Systems (ICS)

• CI are infrastructures that are essential for the functioning of a society and its economy
  – e.g., electricity, gas, telecommunications, water, dams, nuclear plants, public health, transportation, financial services, food...

• CI usually rely on Industrial Control Systems (ICS)
  – different vulnerabilities with respect to regular IT
Industrial Control Systems (ICS)

- the organization also have a regular IT network for administration, sales, etc.
  - … with regular security problems
The Preemptive research project

- Preemptive: “Preventive Methodology and Tools to Protect Utilities”

- focus on cybersecurity of “utilities”: companies managing electricity, water, gas

- objectives
  - prevention and detection
  - methodology and technology
  - final testbed
The Preemptive research project

• Preemptive is founded by the EU (FP7)
  – 12 european (+israel) partners
    (5 research + 7 industry)
  – 6 “end users” (utility operators)
  – three years (ends Feb 2017)

• many results
  – a specific risk assessment methodology
  – many specific IDS/IPS tools

• we focus on the results of uniroma3
ICS Security: specific aspects

• very peculiar technology
  – SCADA-related software
  – embedded system
  – usually not mastered by regular hacker \textcolor{green}{(good)}

• built for safety - not for security
  – not to be resilient to malicious software attacks \textcolor{red}{(bad)}

• old systems, rarely patched/updated \textcolor{red}{(bad)}
  – patching is costly

• elective targets for specific attackers
  – terrorists, opposing governments, intelligence agencies \textcolor{red}{(bad)}
  – much larger resources than regular hackers \textcolor{red}{(bad)}
  – Advanced Persistent Threats, APTs \textcolor{red}{(bad)}
parentesi su malware e APTs
malware

• qualsiasi software che si comporti in modo illecito o malevolo nei confronti dell'utente
• tipicamente associati a un meccanismo di propagazione
  – sociale o tecnologico
• moltissime tipologie e varianti
  – classificazione molto complessa
  – più che una classificazione del software si classificano le tipologie di “comportamento”
    • virus, trojan, worm, rasomware, AdWare, SpyWare, ecc.
    • es. un malware può essere contemporaneamente trojan e virus
propagazione

fonte Microsoft, SIRv11 2011
zombies e botnet

• alcuni malware rimangono in attesa che il sistema sia utilizzato da un hacker (installano una backdoor)
  – tipicamente trojan, virus o worm
• un sistema infetto è detto zombie
• una rete di zombies comandabili coerentemente è detta botnet
• spesso gli zombies sono comandati mediante Internet Relay Chat (IRC botnet)
• usi
  – 50-80% dello spam viene da zombies
    • risparmio di banda, indirizzi diversi confondono gli antispam
  – Distribute DoS (attacchi famosi a Yahoo, eBay, ecc)
  – click frauds (siti con annunci “pay per click”)
  – hosting di siti di phishing

• fonte: http://en.wikipedia.org/wiki/Zombie_computer
Cybercrime Black Market and ecosystem
the market

Fonte: kaspersky (2009)

- botnet: $50 to thousands of dollars for a continuous 24-hour attack.
- Stolen bank account details vary from $1 to $1,500 depending on the level of detail and account balance.
- Personal data capable of allowing the criminals to open accounts in stolen names costs $5 to $8 for US citizens; two or three times that for EU citizens.
- A list of one million email addresses costs between $20 and $100; spammers charge $150 to $200 extra for doing the mailshot.
- Targeted spam mailshots can cost from $70 for a few thousand names to $1,000 of tens of millions of names.
- User accounts for paid online services and games stores such as Steam go for $7 to $15 per account.
- Phishers pay $1,000 to $2,000 a month for access to fast flux botnets
- Spam to optimise a search engine ranking is about $300 per month.
- Adware and malware installation ranges from 30 cents to $1.50 for each program installed. But rates for infecting a computer can vary widely, from $3 in China to $120 in the US, per computer.
market participants - levels

Different Levels of Participants in the Underground Market

Proportion of Participants
Sophistication/skill levels and various roles

Examples:
- Elite researchers
- Exploit developers
- Zero-day researchers
- Malware writers
- Identity collectors
- Programmers
- Tech experts
- As-a-service providers
- Virtual money mule services
- Spammers
- Botnet owners
- Drop service
- Distributors
- Hosted system providers
- Cashiers
- ID/financial data providers
- Buyers
- Observers

RAND - Markets for Cybercrime Tools and Stolen Data, 2014
Advanced Persistent Threats (cyberwar)

• organizzazioni (es. governi) capaci di minacciare continuativamente un obiettivo
  – con mezzi informatici ma non solo

• obiettivi
  – compromissioni di sistemi industriali (stuxnet)
    • primo rootkit per sistemi SCADA
  – reperimento di informazioni (flame)
    • screenshot, voice recording, remote control

• virus sofisticati
  – sfruttamento di vari zero-day threats
  – sfruttamento di collisioni MD5
  – infezioni su varie tecnologie (es. bluetooth, PLC, scada)
Advanced Persistent Threats

peculiarities of APTs
• malware usually operated by very big organizations
• no direct profit but political or market advantages
• leverage insiders for info gathering and initial attack
• knowledgeable
  – about specific industrial processes
  – about deployed countermeasures (e.g. antivirus evasion)
• trade time for stealth (slow attacks)
• based on zero-days
  – e.g. procured on the black market
  – leverage public cloud facilities
famous APTs

• **Stuxnet (2010)**
  – target: iranian uranium enrichment facilities
  – spreads through USB storage and regular IT systems
  – **specifically infects SCADA servers and embedded systems**
    • change control parameters of centrifuges to induce excessive vibration
  – hide from antivirus
  – exploits several new vulnerabilities
  – cryptographic attack

• **others: Duqu (2011), Flame (2012), Duqu 2.0 (2015)**

• apt.securelist.com (kaspersky)
fine parentesi
Integrity techniques for ICS protection and USB security
two “realms”

Regular IT

Industrial Control System
problem setting

• regular IT: considered insecure
• ICS: must be protected from APTs that can easily reach regular IT
• ICS loosely connected
  – USB memory sticks are used
• USB memory are used promiscuously
• USB memory is a spreading vector for APT
idea

• use the Biba integrity model
  – high integrity level: ICS
  – low integrity level: regular IT

• for USB memory, we cannot rely on file system access control
  – why???

filesystem access control is useless

• USB sticks are used promiscuously on untrusted computers (e.g., employee devices)
• access control is not trusted in these devices
• we cannot be sure that nobody tamper with critical data
• hence, we cannot use file system access control
  – we use cryptographic methods: signature
problems for USB filesystem signature/integrity

- **composite data**
  - what about deletion or reverting to previous version of a single file?

- **common approaches**
  - signing each file separately
    - does not detect file deletion/restoration
    - inefficient for large files
  - signing each block separately
    - does not detect restoration of single blocks
  - signing the whole filesystem
    - effective tampering detection
    - highly inefficient: $O(n)$ time for update, $O(n)$ time for check, where $n$ is the total amount of data stored, we aim at have $O(m)$ for update and check, where $m$ is the data read or written
parentesi: merkle hash tree
Authenticated Data Structure (ADS)

• a data structure that speed up hash computation and checks
• useful when
  – the dataset the hash is computed on (n) is large
  – the changed data m are small (m<<n)
  – the retrieved data m are small (m<<n)
• typical hypothesis
  – client of an ADS can keep a hash (constant size) in a trusted environment
  – client of a ADS can use a large amount of untrusted storage
ADS typical usage

• by using an ADS, client can detect tampered data before they are used
  – e.g., before they cause problem in business processes

• typical application
  – cloud storage
    • legal proof of correctness or tampering
    • service level agreement verification
  – backup check
many different ADSes

• Easy example: authenticated list
  – each element e contains an info e.x and a cryptographic hash e.h and pointers e.prev e.next
  – e.h = hash(e.prev.h | e.x)
  – efficiency: append O(1), check O(n)

• Merkle Hash Tree
• Authenticated Skip Lists

• static and dynamic
MHT: how does it work

- a (balanced) tree
- each node \( v \) contain a hash for the data associated with leaves below \( v \)
- client keep only the root hash in a trusted storage

\[
\text{root hash} = V_{0,0} = h( V_{1,0} | V_{1,1} )
\]

\[
V_{1,1} = h( V_{2,2} | V_{2,3} )
\]

\[
V_{2,2} = h(m_3) \quad V_{2,3} = h(m_4)
\]

data must be ordered

\( h(.) \) is a cryptographic hash function
MHT: integrity proof

- proof for $m_i$:
  - consider the path from $m_i$ to root
  - the proof is made of the siblings of the nodes in that path
- example: proof for $m_2$
  - $v_{2,0}$ $v_{1,1}$
- check:
  - assume that client has a trusted version of the root hash (RH)
  - $RH = h(h(v_{2,0} | h(m_2)) | v_{1,1})$
  - compare
    $RH ==$ trusted RH
MHT: check semantic

• client is sure that the data of the reply comes from the dataset associated with the trusted version of the root hash
MHT: efficiency

- for a balanced MHT creating and checking a proof is efficient
- let $n$ the size of the stored data
- length of the proof is $O(\log n)$
MHT: update

- we have to update $m_i$ to a new version $m'_i$
  - root hash will change as well as several internal hashes
- procedure
  - compute proof $p$ for $m_i$ and check it
  - update the hashes of the path to root starting from $m_i$ using content of $p$
  - update trusted root hash
MHT: update

• example: update $m_2$ to a new version $m_2'$

• \( O(\log n) \) time for balanced trees

updated from proof
ADS use case: check of cloud behaviour

- client stores root hash locally
- ADS can be stored in cloud too
- ADS can be applied to regular cloud storage
  - i.e., storage might not know about ADS
ADS authenticated query protocol

**Diagram:**

- **ADS storage**
- **client**
- **Storage**

**Protocol:**

- `AUTH_query(x)`
- `Proof: P_1..P_k`

**Verification:**

- `P_1==H(result)`?
- `HashChain(Proof) == RH`?
ADS authenticated update protocol

- Query ADS x
- Proof
- Update local RH
- Update ADS
- Insert x
security remarks

• tampering with the ADS cannot lead to undetected data tampering
• to break the protection a has collision must be found
• if an ADS is lost, it can be re-created from data
• essentially an ADS is only a speed-up tool
fine parentesi
efficient filesystem integrity

• by using ADS we obtain
  – integrity check that detect any kind of tampering
  – efficiency comparable to any index data structure

• a MHT for integrity of files and directories can be represented by means of files and directories
  – ADS stored in the same USB storage
architecture of the Host Integrity System

- two realms: critical and regular
- only critical machines are equipped with an “Integrity Manager”
  - checks that only genuine data are read
  - write proof that data are genuine
  - based on hash and signature

- USB memory sticks
  - any regular hardware
  - a secure zone is identified (a directory)
  - critical machines can only read from secure zone
special operations

- processes in critical machines read and write USB memory sticks through the **Integrity Manager**
  - redefinition of system call semantic for ADS and root hash handling
other elements

• each host M a private/public key
• Certification Authority (CA)
• root hash is signed by private key and written in the memory stick
  – …along with certificate of M

• possible support of many secure zones
• initial creation of an empty secure zone
architecture

Critical Machine ($M$)

User Processes

User Application

System Processes

Integrity Manager

CA Public Key

M Private Key

$CERT(M)$

Fuse (Linux)

Dokany (Windows)

filesystem driver

USB driver
architecture

Machine ($M$)

System Processes

- Integrity Manager
  - CA Public Key
  - M Private Key
  - $CERT(M)$

File (Linux)
File (Windows)

System driver

USB driver

Secure Zones
Inventory

Secure Zone Z

Accessory Data for Z
- Signature
- Last writer certificate

ADS

Usb Memory stick
gatekeeper

• distributed implementation of the Biba model (no need for networking)
• how to import data/software into the critical realm? special machine: gatekeeper

• gatekeeper
  – like a critical machine but can read any data (and write it into a secure zone)
  – can implement a “complete mediation” for check possibly malicious data before they enter into the critical realm
security remarks

• restoring of a previous backup is not considered an attack

• USB memory stick is considered passive
  – no protection against firmware attacks (unless they show tampered data)
USB firmware attack: BadUSB

• a malicious USB stick declare to be a keyboard
• when inserted into a PC start to “type” commands possibly
  – downloading software (malware)
  – executing software
  – changing configurations
  – typing to create malicious scripts and execute them
protection: USBCheckIn

• it is an hardware that prevents “malicious typing”
• when a USB device pretend to be a keyboard the user is asked to type specific codes
• it is a sort of Captcha for USB devices
USBCheckIn: startup
USBCheckIn: keyboard authorization