Attestation-based Policy Enforcement for Remote Access

Reiner Sailer, Trent Jaeger, Leendert van Doorn, Xiaolan Zhang

IBM Thomas J. Watson Research Center
Hawthorne, NY

Overview

Problem: Remote Access To Corporate Intranets

- Client systems are very dynamic and can exhibit many exploitable vulnerabilities in a typical home environment.
- These vulnerabilities impact corporations through remote access of clients to sensitive corporate data and services.

Solution

- Establish and maintain credibility into client systems during remote corporate access sessions.
- Delegate fine-grained policy enforcement to trusted clients.
- Prototype based on TCG/TPM attestation, Linux Netfilter, IBM Tivoli Access Manager.
Outline

- Problem of State-of-the-art Remote Access
- Establishing Client Credibility
- Client Policy Enforcement
- VPN Policy Representation
Remote Corporate Access

1. Establish Tunnel to Client
2. Check user authorization
3. Enforce Tunneling

Corporate Remote Client

Internet

Corporate Policy Enforcer

Trusted Corporate Intranet
Client Threats / Security Goals

Corporate Remote Client

FW

Internet

VPN Server

Attacker

Trusted Corporate VPN
Client Threats / Security Goals

Security Goal SG1 (Threat 1):
No Client-Firewall Bypass
- Protection against attackers accessing the client tunnel
Client Threats / Security Goals

Client Data Leakage

Attacks 1,2: Client acting on behalf of attacker instead of the authorized user

Security Goal SG1 (威胁 1):
No Client-Firewall Bypass
- Protection against attackers accessing the client tunnel

Security Goal SG2 (威胁 2):
Basic Client Data Containment
- Protection against data leakage (e.g. disable persistent storage)
# Client Threats / Security Goals


<table>
<thead>
<tr>
<th>#</th>
<th>Kernel (SHA1)</th>
<th>Kernel Configuration</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>123AD...A22D1</td>
<td>IMA, Modem, Wireless, Ethernet, Persfw, etc.</td>
<td>NONE</td>
</tr>
<tr>
<td>2</td>
<td>E1F98...34AA1</td>
<td>IMA, Onboard-Eth, Persfw, Netfilter, IPSEC, no other communication, <strong>no modules support, no persistent client storage</strong> (only RAM file system, remote file system)</td>
<td>SG1,SG2</td>
</tr>
<tr>
<td>3</td>
<td>AA424...4131B</td>
<td>Like Kernel #2, but supports persistent ext3 client file system</td>
<td>SG1</td>
</tr>
</tbody>
</table>

### Security Goal SG1:
- **No Client-Firewall Bypass**

### Security Goal SG2:
- **Basic Client Data Containment**
Our Solution

1. Client Policy Enforcement trustworthy?

2a. Client: SG1?/SG2?

2b. Client Policy Enforcer

Packet-level Control

Client

Delegate Enforcement

Network

Client Corporate Intranet

ON / OFF

Corporate Intranet

Attestation & Authorization Service

Corporate Policy Data Base

Client Policy Enforcement trustworthy?

Client Policy Enforcement trustworthy?
Our Solution

**Attestation-based VPN Policy Enforcement**

- **Establish** credibility into integrity of client policy enforcement by initial attestation
  - Trusted path between user, client, and VPN

- **Delegate** client-specific policy enforcement to clients

- **Re-validate** periodically integrity of client policy enforcement by repeated attestation
Our Solution

Advantages

- Client-properties remotely observed throughout remote access session
- Client-specific fine-grained policy efficiently enforced inside the client
- Corporate policy enforcement can focus service-level goals in a less complex and more scalable way
- Access control policies for a large set of heterogeneous clients can be managed efficiently
Outline

- Problem of State-of-the-art Remote Access
  - Establishing Client Credibility
  - Client Policy Enforcement
  - VPN Policy Representation
Establishing Client Credibility

Measurements

Remote Access Client

Data

Program

Boot-Process

Kernel

Kernel module

System-Representation

TPM-Signed PCR Integrity Value

Analysis

Client

VPN Server

Client Properties

ext. Information (CERT,...)

Known Fingerprints

Example – Client Property Measurements

<table>
<thead>
<tr>
<th>CLIENT Measurement List $E(\text{client})$</th>
<th>VPN Fingerprint DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>#000: BC55F0AFE013C3402F00E0AA11EE6CFAA2B4D2AB</td>
<td>boot_aggregate (bios and grub stages)</td>
</tr>
<tr>
<td>#001: A8A865C7203F2565DDEB511480B0A2289F7D035B</td>
<td>grub.conf (boot configuration)</td>
</tr>
<tr>
<td>#002: 1238AD50C652C88D139EA2E9987D06A99A2A22D1</td>
<td>vmlinuz-2.6.5-bk2-lsmtcg</td>
</tr>
<tr>
<td>#003: 84ABD2960414CA4A448E0D2C9364B4E1725BDA4F</td>
<td>init (first process)</td>
</tr>
<tr>
<td>#004: 9ECF02F90A2EE2080D4946005DE47968C8A1BE3D</td>
<td>ld-2.3.2.so (dynamic linker)</td>
</tr>
<tr>
<td>#005: 336536B0E22FF762BB539D7FCB7CD283D4622342</td>
<td>libc-2.3.2.so</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>#439: 2300D59E0AB01A6B9D203CE2A7655177E6247882</td>
<td>persfw_user (client policy agent)</td>
</tr>
<tr>
<td>#440: BB18CB801C9D27E255C209CB56A47C9EA9CBDD12</td>
<td>libpdauthzn.so (policy client libraries)</td>
</tr>
<tr>
<td>#441: D12D96BAA8D148BC3C8DF0F3B758S9B425A829EE</td>
<td>libpdcore.so</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>#453: DF541AEDFECB35116808306E89C05591E3ABE160</td>
<td>local.conf (policy agent)</td>
</tr>
<tr>
<td>#454: 6AC585D072AC9F32AC9CDF8698CAA004AA6DC781</td>
<td>authzn_persfw.db (policy db)</td>
</tr>
</tbody>
</table>

\[
\text{Known Fingerprints} = \text{Acceptable} \cup \text{Local} \cup \text{Remote} \cup \text{Uncontrolled} \cup \text{Malicious}
\]
Establishing Client Credibility

\[
\text{client} \in \textbf{IntegrityHigh} \iff \forall e \in \mathcal{E}(\text{client}): \\
\quad (e \in \text{Acceptable})
\]

\[
\text{client} \in \textbf{IntegrityMedium} \iff \text{client} \notin \textbf{IntegrityHigh} \\
\quad \forall e \in \mathcal{E}(\text{client}): (e \in \text{Acceptable} \cup \text{Local})
\]

\[
\text{client} \in \textbf{Distrusted} \iff \exists e \in \mathcal{E}(\text{client}): \\
\quad (e \notin \text{Known}) \lor (e \in \text{Malicious} \cup \text{Uncontrolled} \cup \text{Remote})
\]
Establishing Client Credibility

- Determine securely all software loaded on the client (load-guarantees)
  - Kernel, libraries, applications, relevant configuration files

- Establish credibility into
  - client integrity
  - client policy enforcement
    (policy agent, policy database, client packet filter)
Outline

- Problem of State-of-the-art Remote Access
- Establishing Client Credibility
  - Client Policy Enforcement
  - VPN Policy Representation
Client Policy Enforcement

1. Client Policy Enforcement trustworthy?

2a. Client: SG1?/SG2?

2b. Packet-level Control

Client Policy Enforcer

Delegate Enforcement

- Attestation & Authorization Service
- Corporate Policy Data Base
- Corporate Intranet

Network

ON / OFF
Client Policy Enforcement

\[
[\text{decision, client-constraints, packet-constraints}] := \\
\text{AM}_\text{azn}(\text{capabilities}(\text{userid}), \text{service}, \text{direction});
\]

1. **decision**
   - denied/
   - permitted
   - (VPN)

2a. **Get Client Kernel Measurement**
   - Look-up Known Properties
   - Client Policy Agent
   - true/
   - false

2b. **Stateful Packet Filter**
   - Applied directly to pattern/state matching

<table>
<thead>
<tr>
<th>#</th>
<th>Kernel SHA1</th>
<th>Configuration</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>E1F9...4AA1</td>
<td>...(see paper)</td>
<td>SG1 &amp; SG2</td>
</tr>
<tr>
<td>2</td>
<td>AA42...131B</td>
<td>...(see paper)</td>
<td>SG1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Client Policy Enforcement

nf_register_queue_handler(PF_INET, do_persfw_policy, pdRequestQueue);

Check Policy (low overhead)

- TCP?
  - No
  - Yes
    - "reinsert(packet, ACCEPT)"
    - "reinsert(packet, DROP)"

- SYN && !ACK?
  - Yes
    - "add_conn(conn, invariant)"
  - No
    - OK && Invariant
      - "reinsert(packet, ACCEPT)"
    - ELSE
      - "reinsert(packet, DROP)"

Netfilter

FW/Packet

AuthorizationResponse (11 000/s): 
{"permitted", Constraints=[SG1 Required, …]}
Outline

- Problem of State-of-the-art Remote Access
- Establishing Client Credibility
- Client Policy Enforcement
- VPN Policy Representation
VPN Policy Representation

Corporate Policy Data Base

Constraints

HTTPS outgoing – ALLOW
ClientIntegrity: Med
SG1: Required
SG2: Not Required
Protocol: TCP/443
ServerIP: 10.9.*.*
VPN
PA
PA
FW
FW
VPN Policy Representation

Authorization-Request: { user credentials & “X” & 
"/persfw/corporate/appservices/https/out"}

-/persfw acl {[T, anybody]}
-/persfw/corporate acl {[T, anybody]}
-/persfw/corporate/appservices acl {[T, anybody]}
-/persfw/corporate/appservices/https acl {[T, group https_out]; [T, group https_in]}
-/persfw/corporate/appservices/https/out acl {[X, group https_out]}

(Retum = “permitted”) ≡ (user ∈ https_out)

1. allow (user, https, out) → add
2. add client enforced constraints

ClientIntegrity: Medium
Security Goal 1: Required
Security Goal 2: Not Required
Protocol: TCP / 443
ServerIP: 10.9.*.*
Performance

- **VPN Server: Client Property Determination**
  - 1-5 Seconds depending on client software-stack complexity (non-optimized, can be delegated to edge access servers)

- **Client Policy Retrieval**
  - User space policy retrieval: ~850 microseconds, 1100/s (tested with UDP traffic and no policy cache, sub-optimal)

- **Client Policy Enforcement – using policy cache**
  - 4% penalty on TCP stream traffic (packet filter overhead)
  - 11% on UDP traffic (classify, policy, no tracking)

- Negligible overhead for most applications
Summary: Protecting Intranets From Client Vulnerabilities

Attestation-based Remote Policy Enforcement

- **Establish** credibility into client (trusted path):
  - Client run-time integrity
  - Client policy enforcement capabilities

- **Delegate** client-specific policy enforcement to clients
  - Coarse-grained **client run-time property enforcement** (e.g., Security Goals 1,2)
  - Fine-grained **packet-level enforcement** (filtering)

- **Monitor periodically** client credibility regarding run-time and policy enforcement (repeated efficient attestation)

Further Information – E-mail: sailer@us.ibm.com