Automated Run-Time Validation for Cryptographic Modules

NIST ITL Computer Security Division
Apostol Vassilev, Harold Booth

Cisco Security & Trust Organization (STO)
Barry Fussell, David McGrew, Sam Farthing

May 2016
Traditional Conformance Testing

Vendor

*Designs and Produces*
- Hardware • Software • Firmware
- Define Boundary
- Define Approved Mode of Operation
- Security Policy

CST Lab

*Tests for Conformance*
- Derived Test Requirements
- CAVP Algorithm Testing
- Documentation Review
- Source Code Review
- Operational and Physical Testing

CMVP

*NIST and CSEC*
- Validates
- Review Test Results
- Ongoing NVLAP Assessment
- Issue Certificates
- NIST Cost Recovery Fee

User

*Specifies and Purchases*
- Security and Assurance
- Applications or products with embedded modules
A look at the challenges today: key facts in the Verizon 2016 Breach Investigation Report

1. **Most attacks exploited known vulnerabilities where a patch has been available for months, often years**
   - The top 10 vulnerabilities [Common Vulnerabilities and Exposures, or CVEs] accounted for 85% of successful exploit traffic.

2. **Main reason**
   - 58% of businesses don’t have “mature” patch management processes.

3. **No one is immune**
   - Attackers are getting better
   - In 93% of cases, it took attackers minutes or less to compromise systems.

4. **Most breaches are about money**
   - Any cyber asset you have that can be monetized makes you a target.
   - Cyber-espionage targets IP in manufacturing [47% of breaches] and IT.
Limitations of Traditional Conformance Testing

1. **Long Validation Cycles**
   - Well beyond product development cycles
   - Hinder adoption of new technology by the Federal Government

2. **Shallow Depth of Testing**
   - Software testing methodology inadequate for today’s complexity of crypto implementations
   - Hardware testing lags state-of-the-art

3. **Costly and Rigid**
   - Difficult to obtain compliance assurance on platforms of actual use
   - Limits the industry’s efforts to validate more products
   - Prevents the industry from fixing critical problems, e.g. CVE, without breaking program rules

4. **Impossible to fix within the existing box**
   - Some improvements help but fall short of solving the problems
Government-Industry Working Group

1. **Announced last year**
   - Cybersecurity Innovation Forum, September 2015, Washington DC
   - ICMC15, November 2015, Rockville MD

2. **Initiated in December 2015**
   - Founding face-to-face meeting at NIST

3. **Goals**
   A. **Provide powerful economic incentives to the industry**
      - Reduce the length and cost of validations
      - Enable Just-In-Time validations through verifiable test artifacts
   B. **Widen agencies’ access to latest technologies**
      - Maintaining FISMA compliance with up-to-date patches
   C. **Align with other government test programs**
      - Enable Common Criteria to consume FIPS 140 validations not just CAVP
Government-Industry Working Group Structure
Working Group Status

1. **Scoping work**
   - Finalizing charter documents for each work area
   - Analysis of FIPS 140 DTR’s applicable to work areas

2. **First milestone reached**
   - Automated Cryptography Validation Protocol - ACVP
   - Automated Algorithm Validation System
   - Test and validation automation demo
Automated Cryptography Validation Protocol
Considerations

- An automated approach is needed for scalability
- On premise or via a validation authority
- Customers can test deployed equipment
- Phased delivery
- Version control to support new requirements and backward compatibility
- Automated revalidation eliminates tension between patches and validation
- Validation Authority (NIST)
- Expedite revalidation
- Mitigate threats as late in the software life cycle as possible
- Help address Insider Threats
- Simple Protocol can work with any crypto implementation that has access to HTTP
- 3rd Party Verification
- Software Lifecycle
- Product Coverage
- Automation
- Remote Validation
- Extensibility
- Reduce Tension
ACVP testing can be performed from test cycle to runtime environment.
Software Maintenance Life Cycle

Initial Deployment

Vulnerability found

Vulnerability fixed

FIPS re-validation cycle

MR Deployment

FIPS re-validation cycle

With ACVP

Initial Deployment

Vulnerability found

Vulnerability fixed
ACV Protocol
“Lightweight standards track protocol built on top of existing standard protocols and encoding.”

- TLS 1.2 (optional)
- HTTP(S)
- Java Script Object Notation (JSON)
- JSON Web Token (JWT) Authorization
- Protocol does not define authentication method
ACVP Architecture
Proxy/Validation Authority Architecture
Automated Cryptographic Validation System

**ACV Proxy/Server:**
- Web hosted service
- Interacts with NIST ACV Server to obtain JSON KAT data
- Optionally generates JSON test vectors
- Optionally performs results verification
- Reports JSON KAT results to NIST ACV Server

**Validation Authority Server:**
- Web hosted service w/ REST API
- Registers ACV Servers
- Generates JSON KAT vectors
- Validates JSON KAT results
- Publishes validation results from trusted vendor ACV Servers

**ACV Client:**
- Integrated into Device under test
- May convert JSON test vectors to format acceptable by crypto module under test
- Returns KAT answers to ACV server in JSON format

**ACV Protocol**

**Vendor ACV Server**

**Device Under Test**

**Crypto Module**

**Seed**

**JSON Test Vectors**

**Responses**

© 2016 Cisco and/or its affiliates. All rights reserved. Cisco Public
Base Architecture
Automated Cryptography Validation Protocol

**ACV Server:**
- Web hosted service
- Generates JSON test vectors
- Performs results verification

**ACV Protocol:**
- Standards-based protocol
- Developed in partnership w/ CMVP
- Extensible to mitigate additional vectors over time
- Open Source to enable independent verification

**ACV Client:**
- Integrated into Device under test
- May convert JSON test vectors to format acceptable by crypto module under test
- Returns KAT answers to ACV server in JSON format

**Entropy Source**
- DRBG

**ACV Protocol**
- Test Vectors
- Responses

**Crypto Module**
- Encryption
- Authentication
- Key Establishment
- Signatures

**Device Under Test**
- Public Key Generation

**ACV Server:**
- Web hosted service
- Generates JSON test vectors
- Performs results verification

**ACV Protocol:**
- Standards-based protocol
- Developed in partnership w/ CMVP
- Extensible to mitigate additional vectors over time
- Open Source to enable independent verification
ACVP Flows

1. Client Registration and Vector Set Creation
2. Client GET of Vector Sets
3. Client POST of Known Answer Test Results
4. Client uses GET method to retrieve results
ACVP Flows – Client Registration and Vector Set Creation

Device Under Test

- ACV Server
- ACV Client
- Crypto Module

- http(s) session establishment
- POST Operating Environment
  And Crypto Capabilities
- ACV Server Responds with a list of vector set IDs
Example: Client registration json body

{
    "operation": "register",
    "acv_version": "0.2",
    "oe_information": {
        "vendor_name": "Cisco", "vendor_url": "www.cisco.com",
        "contact": "John Doe", "contact_email": "johndoe@cisco.com",
        "module_name": "Cisco ACV Test Module 1.0",
        "module_type": "Software",
        "operational_environment": {
            "module_version": "1.0",
            "processor": "Intel Woodcrest",
            "operating_system": "Linux 3.1"
        }
    }
}
Example: Client registration json body

"implementation_description": "Sample crypto module for demonstrating ACV protocol."

},
"capability_exchange": {
"algorithms": [
{
"algorithm": "AES-GCM",
"mode": "both", "iv_gen": "internal", "iv_gen_mode": "8.2.1",
"key_lens": [ 128, 256], "tag_lens": [ 96], "iv_lens": [ 96],
"pt_lens": [ 0, 128, 136 ], "aad_lens": [ 128, 136 ]
}
]
Example: Server registration response

```json
{  "acv_version" : 0.2 ,

    "capability_response" : 

    {  "vector_sets" : [{"vs_id":1457},{"vs_id":1458}]

    },

    "test_session" : {"test_id":474} ,

    "access_token" : "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiAibmlzdC5vcmciLCJleHAiOiAxNDYzMDc4NTI4LCJjb21wYW55IjogIkNpc2NvIiwianRpIjo0NzR9.6AggT4h10YGrNiTlZrf-I7jend5ut6V9Ai_or_LohJE"

}
```
ACVP Flows – Client GET of Vector Sets

ACV Server Responds with vectors sets prepared for vs_id=1459
Example: Partial vector set in json format

```json
{
"acv_version": "0.2", "vs_id": 1459, "algorithm": "AES-GCM",
"mode": "encrypt",
"test_groups": [ 
{
"keylen": 128, "ivlen": 96, "ptlen": 0, "aadlen": 128, "taglen": 128,
"tests": [
{
"tc_id": 23400,
"key": "F3EE815AD9AD595BE7BBE574B25D6313", "pt": "",
"aad": "A01834F0C26F092BADEB6A49A5D598E8"
},
{
"tc_id": 23401,
"key": "0498D606AF65E389157215125147782C",
```
ACVP Flows – Client POST of Known Answer Test Results

http(s) session establishment

POST /validation/acvp/vectors?vs_id=1459

ACV Server Responds with 200 OK
Example: Partial vector set response in json format

```json
{
  "acv_version": "0.2",
  "vs_id": 1459,
  "algorithm": "AES-GCM",
  "mode": "encrypt",
  "test_results": [
    {
      "tc_id": 23400,
      "iv": "01020304A685AE2E7E17642B",
      "ct": "",
      "tag": "D0550D9C53F676C027B998B3C5E5C3C2"
    }
  ]
}
```
ACVP Flows – Client uses GET method to retrieve results

Device Under Test

ACV Server

http(s) session establishment

ACV Client

Crypto Module

GET /validation/acvp/results?vs_id=1459

ACV Server Responds with 200 OK and json formatted results for vs_id=1459
Example: Results returned in json format

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Thu, 18 May 2016 20:01:26 GMT

{"acv_version": "0.2",
 "results" : {
   "disposition" : "passed",
   "1462" : "passed" }
}
Q & A
Example: Decode of Access Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpc3MiOiAibmlzdC5vcmciLCJleHAiOiA
xNDYzMDc4NTI4LCJjb21wYW55IjogIkNpc2M
vIiwianRpIjo0NzR9.6AggT4h10YGrNiTiZr
f-I7jend5ut6V9Ai_or_LohJEeyJhbGciOiJIU
zI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIx
MjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9I
IiwYWRtaW4iOnRydWVG.TJVA950rM7E2cBa
b30RMHrHDCeFxjoYZgeF0NFh7HgQ

HEADER: ALGORITHM & TOKEN TYPE

{
    "alg": "HS256",
    "typ": "JWT"
}

PAYLOAD: DATA

{
    "iss": "nist.org",
    "exp": 1463078528,
    "company": "Cisco",
    "jti": 474
}