Steel: Scaling up verification of realistic programs

AYMERIC FROMHERZ
A need for verified software

• Many security vulnerabilities are due to bugs in implementations
A need for verified software

• Many security vulnerabilities are due to bugs in implementations

Issue 1804: cryptoapi: SymCrypt modular inverse algorithm
Reported by taviso@google.com on Tue, Mar 12, 2019, 9:15 PM PDT  Project Member

There's a bug in the SymCrypt multi-precision arithmetic routines that can cause an infinite loop when calculating the modular inverse on specific bit patterns with bcryptprimitives!SymCryptFde!Mod!Inv!Generic.

I've been able to construct an X.509 certificate that triggers the bug. I've found that embedding the certificate in an S/MIME message, authenticode signature, schannel connection, and so on will effectively DoS any windows server (e.g. ipsec, iis, exchange, etc) and (depending on the context) may require the machine to be rebooted. Obviously, lots of software that processes untrusted content (like antivirus) call these routines on untrusted data, and this will cause them to deadlock.
A need for verified software

• Many security vulnerabilities are due to bugs in implementations

Issue 1804: cryptoapi: SymCrypt modular inverse algorithm
Reported by taviso@google.com on Tue, Mar 12, 2019, 9:15 PM PDT

There’s a bug in the SymCrypt multi-precision arithmetic routines that can cause an infinite loop when calculating primitives!SymCryptFsdefModInvGeneric.

I’ve been able to construct an X.509 certificate that triggers the bug. I’ve found that embedding the certificate in an S/MIME message, authenticode signature, schannel connection, and so on will effectively DoS any windows server (e.g. ipsec, iis, exchange, etc) and (depending on the context) may require the machine to be rebooted. Obviously, lots of software that processes untrusted content (like antivirus) call these routines on untrusted data, and this will cause them to deadlock.
A need for verified software

• Many security vulnerabilities are due to bugs in implementations

Issue 1804: cryptoapi: SymCrypt modular inverse algorithm

Reported by taviso@google.com on Tue, Mar 12, 2019, 9:15 PM PDT  Project Member

There's a bug in the SymCrypt multi-precision arithmetic routines that can cause an infinite loop when calculating
bcryptprimitives!SymCryptFdefModInvGeneric.

I've been able to construct an X.509 certificate that triggers the bug. I've found that embedding the certificate in an S/MIME message, authenticating the connection, and so on will effectively DoS any windows server (e.g. ipsec, iis, exchange, etc) and (depending on the context) may require the server to reset. Obviously, lots of software that processes untrusted content (like antivirus) call these routines on untrusted data, and this will cause them to do...
A need for verified software

• Many security vulnerabilities are due to bugs in implementations

**Issue 1804: cryptoapi: SymCrypt modular inverse algorithm**

Reported by taviso@google.com on Tue, Mar 12, 2019, 9:15 PM PDT  Project Member

There's a bug in the SymCrypt multi-precision arithmetic routines that can cause an infinite loop when calculating bCrypt primitives!SymCryptFdefModInvGeneric.

I've been able to construct an X.509 certificate that triggers the bug. I've found that embedding the certificate in an S/MIME message, authenticating a connection, and so on will effectively DoS any Windows server (e.g. IIS, exchange, etc) and (depending on the context) may require the attacker to have already compromised the server. Obviously, lots of software that processes untrusted content (like antivirus) call these routines on untrusted data, and this will cause them to crash.

• Critical software requires the highest security guarantees
A need for verified software

- Many security vulnerabilities are due to bugs in implementations

Issue 1804: cryptoapi: SymCrypt modular inverse algorithm

Reported by taviso@google.com on Tue, Mar 12, 2019, 9:15 PM PDT

There's a bug in the SymCrypt multi-precision arithmetic routines that can cause an infinite loop when calculating brcryptprimitives!SymCryptDefModInvGeneric.

I've been able to construct an X.509 certificate that triggers the bug. I've found that embedding the certificate in an S/MIME message, authenticating the connection, and so on will effectively DoS any windows server (e.g. ipsec, iis, exchange, etc) and (depending on the context) may require the server to shut down.

Obviously, lots of software that processes untrusted content (like antivirus) call these routines on untrusted data, and this will cause them to do the same thing.

- Critical software requires the highest security guarantees
- Bug finding: Not good enough
A need for verified software

- Many security vulnerabilities are due to bugs in implementations
- Critical software requires the highest security guarantees
- Bug finding: Not good enough
- Preventing buffer overflows: A good first step, but still insufficient
A need for verified software

- Many security vulnerabilities are due to bugs in implementations
- Critical software requires the highest security guarantees
- Bug finding: Not good enough
- Preventing buffer overflows: A good first step, but still insufficient
- We need **formal verification!**
Formal verification today

We can verify small industrial-grade systems
(Cryptographic providers,
Parsers and Serializers, ...)

Formal verification today

We can verify small industrial-grade systems
(Cryptographic providers,
Parsers and Serializers, ...)

EVERCRYPT
Formal verification today

We can verify small industrial-grade systems
(Cryptographic providers,
Parsers and Serializers, ...)
Formal verification today

We can verify small industrial-grade systems
(Cryptographic providers,
  Parsers and Serializers, ...)

But
• Time consuming, and requires expert knowledge
• Does not scale to large systems, or to realistic concurrent programs
Steel: Efficient verification of general-purpose, concurrent systems
Steel: Efficient verification of general-purpose, concurrent systems

Specification
(“the mathematical truth”)
Steel: Efficient verification of general-purpose, concurrent systems

Specification ("the mathematical truth")

proof

Steel ("C-like")
Steel: Efficient verification of general-purpose, concurrent systems

- Specification ("the mathematical truth")
- Steel ("C-like")
- Extracts to
- C code (.c, .h)

proof
Steel: Efficient verification of general-purpose, concurrent systems

Security Theorems

Proof

Specification ("the mathematical truth")

Proof

Steel ("C-like")

Extracts to

C code (.c, .h)
Memory reasoning is hard
Memory reasoning is hard

Node* x = 1 → 2 → 1 → 3 → NULL

Node* y = 4 → NULL
Memory reasoning is hard
Memory reasoning is hard

```
Node* x
  1 -> 2 -> 1 -> 3 -> NULL
```

```
Node* y = 4 -> NULL
```
Memory reasoning is hard

Node* x
  1 --> 2 --> 1 --> 3 --> NULL

Node* y = 4 --> NULL
Steel: Core ideas
Steel: Core ideas

• Semi-automated memory reasoning, using a new take on separation logic
Steel: Core ideas

• Semi-automated memory reasoning, using a new take on separation logic

• Programs are always safe, with user-controlled verification
Steel: Core ideas

• Semi-automated memory reasoning, using a new take on separation logic
• Programs are always safe, with user-controlled verification
• Permissions on objects: Exclusive mutable or shared read-only access
Steel: Core ideas

- Semi-automated memory reasoning, using a new take on separation logic
- Programs are always safe, with user-controlled verification
- Permissions on objects: Exclusive mutable or shared read-only access
- Fork/Join concurrency with locks
Steel: Preliminary results
Steel: Preliminary results

- Efficient reasoning about complex data structures (e.g. doubly-linked lists)
Steel: Preliminary results

- Efficient reasoning about complex data structures (e.g. doubly-linked lists)
- Reimplementation and verification of cryptographic primitives
  - Verification time is smaller
  - Specifications are smaller and more readable
Our vision for Steel
Our vision for Steel

• Security guarantees for concurrent protocols (e.g. QUIC)
Our vision for Steel

• Security guarantees for concurrent protocols (e.g. QUIC)

• Verification of critical system components in Microsoft’s cloud platform (Azure)
Our vision for Steel

- Security guarantees for concurrent protocols (e.g. QUIC)
- Verification of critical system components in Microsoft’s cloud platform (Azure)
- Rust frontend for Steel
Our vision for Steel

• Security guarantees for concurrent protocols (e.g. QUIC)

• Verification of critical system components in Microsoft’s cloud platform (Azure)

• Rust frontend for Steel

End-to-end security guarantees for large, industrial-grade systems will become more feasible
Our vision for Steel

- Security guarantees for concurrent protocols (e.g. QUIC)
- Verification of critical system components in Microsoft’s cloud platform (Azure)
- Rust frontend for Steel

End-to-end security guarantees for large, industrial-grade systems will become more feasible

fromherz@cmu.edu