MSRPC Auditing Tools and Techniques

DeepSec 2007

Aaron Portnoy [1] Cody Pierce [2]

[1] aportnoy@tippingpoint.com

[2] cpierce@tippingpoint.com

DeepSec Fall 2007

About Us

- Work at TippingPoint's Digital Vaccine Labs
  - Responsible for vuln-dev, patch analysis, pen-testing
  - Keep tabs on us at http://dvlabs.tippingpoint.com
- Authors and contributors to:
  - Sulley Fuzzing Framework
  - PyEmu x86 Emulator
  - PaiMei Reverse Engineering Framework

Talk Outline

- Background
  - Why do we care about MSRPC in 2007?
- History of MSRPC
  - Issues and Mitigations
  - How it Works
- RPC Tools
  - Existing Tools
- Problems Auditing
  - Locating MSRPC Services
  - Talking to MSRPC Services
- What We've Done
  - Our Toolset
- Demos

Why do we care about MSRPC in 2007?

- Simple bugs are still turning up
  - MS07-029 Vulnerability in Windows DNS RPC Interface
  - MS06-070 Vulnerability in Workstation Service
  - MS06-066 Vulnerabilities in Client Service for NetWare
  - MS06-040 Vulnerability in Server Service
  - MS06-025 Vulnerability in Routing and Remote Access
- 3rd parties still implement obscenely unsafe RPC services
  - Computer Associates (BrightStor Message/Tape Engine
- Many MSRPC services haven't been fully audited
- Trying to audit 3rd party RPC is still tedious
  - Takes longer to get the NDR correct than to find bugs

History of MSRPC

- MSRPC Vulnerabilities
  - Some DoS bugs as far back as 1998
  - MS00-066 Malformed RPC Packet Vulnerability
  - MS03-026 Buffer Overrun In RPC Interface (Blaster Worm)
  - MS04-031 Vulnerability in NetDDE
  - MS05-047 Vulnerability in Plug and Play
  - MS07-029 Vulnerability in Windows DNS RPC Interface
  - ad nauseam
- Other Issues
  - Interface hopping, NULL sessions
- Some Architectual Mitigations
  - Named pipe firewalls
  - NULL session restrictions

Current Issues

- Despite architectual changes to MSRPC
  - 3rd parties still do bad things.. often
- Examples
  - Google for 'CA brightstor rpc vulnerability' (32,600 hits currently)
  - Samba's recent heap overflows
  - Novell Client Print Services RPC Stack Overflows

MSRPC - Why it was developed

- Client/Server model for remotely calling functions as if they were local
- Microsoft forked off from the DCE standard
  - MSRPC has support for Unicode strings, more complex size calculations, interface inheritance
- Some Microsoft services that utilize MSRPC
  - Print Services
  - Message Queuing
  - DNS
  - Exchange
  - Distributed File System
  - Workstation Services

MSRPC - How it Works

- UUIDs
  - UUIDS are unique identifiers for a given interface
- Stubs
  - Allow for a client to call a function in the stub locally but have it executed remotely
  - midl.exe from Microsoft generates these
- IDL Files
  - Defines the interfaces, structures, functions and their arguments
    - Structures
    - Unions
    - Opcodes (functions)

MSRPC - How it Works (cont.)

- Communication (endpoints)
  - TCP
  - UDP
  - SMB (remote named pipes)
  - local named pipes
  - HTTP
  - Other more obscure protocols...
- The endpoint mapper (EPM)
  - Query the endpoint mapper to determine port information for a given UUID
  - Only works if the service registers itself with the EPM

MSRPC - Binaries

- RPC information is stored in the binaries using the following important data and structures
  - Format String
    - Defines the parameter, data structures, and return values pulled from the IDL
  - RPC_SERVER_INTERFACE structure
    - the TransferSyntax element defines the UUID
    - the InterpreterInfo element points to the MIDL_SERVER_INFO structure
  - MIDL_SERVER_INFO
    - DispatchTable element points to the opcode handler functions

Problems Auditing RPC

- First Steps
  - Locating the module that defines the server
    - Could be in the main EXE or any loaded DLLs
  - Retrieving IDL information
    - Sometimes the tools used return incomplete information
  - Determining pipe names
    - Often requires reversing the binary in IDA
    - 3rd party tools can aid in this
  - Determining authentication requirements

Problems Auditing RPC

- Creating the request
  - NDR marshalling is complicated
    - If you are one byte off, communication will likely fail
- Communicating with the server
  - Unmarshalling problems
    - Bad Stub Data
    - Debugging this error can be very trying
  - Domain or computer name requirements
    - Some services validate these and will bail if not correct

RPC Tools

- unmidl.py
  - Written by Dave Aitel
    - Hi, Dave
    - Based off muddle, retrieves IDL information from the format strings in the binaries
    - Better handles complex objects
    - http://www.immunitysec.com/resources-freesoftware.shtml

RPC Tools

- mIDA
  - MIDL Analyzer for IDA
  - Written and maintained by Tenable Security
  - IDApython script
  - Also pulls out IDL information from binaries
  - http://cgi.tenablesecurity.com/tenable/mida.php

RPC Tools

- rpcdump
  - Many tools by this name
    - Microsoft
    - CORE Security
    - Sir Dystic
  - Tool for enumerating endpoint information
  - Queries the endpoint mapper
- ifids
  - Dumps interface information given a protocol sequence and a host

Toolset Components

- PyMSRPC consists of the following components
- Lexer and Parser
  - Allows us to skip the process of rewriting mIDA or unmidl
- A library of NDR objects
  - Defines marshalling information for NDR types
  - Allows for retrieval of the on-the-wire bytestream
- Utlizes Impacket from CORE for transport
  - We extend it's functionality to support context handle communication
- Tie-ins for the Sulley Fuzzing Framework
  - Allows you to fuzz RPC services parsed by PyMSRPC

Using PyMSRPC

- You simply provide an IDL file
- You will be returned:
  - Instantiated, linked python objects for each:
    - UUID
    - Opcode
    - Structure
    - Union
    - Array
  - You can then populate them with data (optionally) and instantly communicate with the server
- PyMSRPC takes a couple of seconds to parse the largest Microsoft IDL (2000 lines) and give you these objects

Why is this useful?

- You no longer have to worry about the complicated and error prone marshalling process
  - See next slide
- You can immediately communicate and audit an RPC service

Why is this useful?

- What follows is the example bytestream you would have to write by hand without our toolset:

\x44\x43\x42\x41\x05\x00\x00\x00\x00\x00\x00\x00\x05
\x00\x00\x00\x54\x00\x45\x00\x53\x00\x54\x00\x00\x00
\xaa\xaa\x44\x43\x42\x41\x05\x00\x00\x00\x00\x00\x00
\x00\x05\x00\x00\x00\x54\x00\x45\x00\x53\x00\x54\x00
\x00\x00\xaa\xaa\x44\x43\x42\x41\x05\x00\x00\x00\x00
\x00\x00\x00\x05\x00\x00\x00\x54\x00\x45\x00\x53\x00
\x54\x00\x00\x00\xaa\xaa\x02\x00\x00\x00\x02\x00\x00
\x00\x44\x43\x42\x41\x02\x00\x00\x00\x44\x43\x42\x41
\x02\x00\x00\x00\x44\x43\x42\x41\x44\x43\x42\x41\x02
\x00\x00\x00\x02\x00\x00\x00\x02\x00\x00\x00\x02\x00
\x00\x00\x44\x43\x42\x41\x44\x43\x42\x41\x44\x43\x42
\x41\x02\x00\x00\x00\x02\x00\x00\x00\x02\x00\x00\x00

Why is this useful?

\x02\x00\x00\x00\x44\x43\x42\x41\x05\x00\x00\x00\x00
\x00\x00\x00\x05\x00\x00\x00\x54\x00\x45\x00\x53\x00
\x54\x00\x00\x00\xaa\xaa\x05\x00\x00\x00\x00\x00\x00
\x00\x05\x00\x00\x00\x54\x00\x45\x00\x53\x00\x54\x00
\x00\x00\xaa\xaa\x05\x00\x00\x00\x00\x00\x00\x00\x05
\x00\x00\x00\x54\x00\x45\x00\x53\x00\x54\x00\x00\x00
\xaa\xaa\x05\x00\x00\x00\x00\x00\x00\x00\x05\x00\x00
\x00\x54\x00\x45\x00\x53\x00\x54\x00\x00\x00\xaa\xaa
\x05\x00\x00\x00\x00\x00\x00\x00\x05\x00\x00\x00\x54
\x00\x45\x00\x53\x00\x54\x00\x00\x00\xaa\xaa\x05\x00
\x00\x00\x00\x00\x00\x00\x05\x00\x00\x00\x54\x00\x45
\x00\x53\x00\x54\x00\x00\x00\xaa\xaa\x02\x00\x00\x00
\x44\x43\x42\x41\x02\x00\x00\x00\x00\x00\x00\x00\x00

Why is this useful?

\x02\x00\x00\x00\x44\x43\x42\x41\x05\x00\x00\x00\x00
\x21\x01\x00\x05\x00\x00\x00\x54\x00\x45\x00\x53\x00
\x54\x00\x02\x00\xaa\xaa\x01\x00\xaa\x00\x00\x00\x00
\x00\x05\x00\x00\x00\x54\x01\x45\x00\x53\x00\x54\x00
\x00\x00\xaa\xaa\x05\x00\x00\x00\x00\x00\x00\x00\x05
\x00\x00\x00\x61\x10\x45\x00\x53\xaa\x54\x00\x00\x00
\xaa\xaa\x05\x00\x01\x00\x10\x00\x00\x00\x05\x00\x00
\x00\x54\x00\x45\x01\x53\x10\x54\xaa\x00\x00\xaa\xaa
\x15\x00\x00\x00\x01\x00\x10\x00\x05\x00\x00\x00\x54
\x00\x45\x00\x53\x01\x54\x00\x00\x00\xaa\xaa\x05\x00
\x00\x00\x00\x00\x00\x00\x05\x00\x00\x00\x54\x00\x45
\x00\x53\x00\x61\x00\x00\x00\xaa\xaa\x02\x00\x00\x00
\x46\x43\x42\x41\x02\x00\x00\x00\x00\x00\x00\x00\x00

Why is this useful?

\x02\x00\x00\x00\x44\x43\x42\x41\x05\x00\x00\x00\x00
\x00\x00\x00\x05\x00\x00\x00\x54\x00\x45\x00\x53\x00
\x54\x00\x00\x00\xaa\xaa\x05\x00\x00\x00\x00\x00\x00
\x00\x05\x00\x00\x00\x54\x00\x45\x00\x53\x00\x54\x00
\x00\x00\xaa\xaa\x05\x00\x00\x00\x00\x00\x00\x00\x05
\x00\x00\x00\x54\x00\x45\x00\x53\x00\x54\x00\x00\x00
\xaa\xaa\x05\x00\x00\x00\x00\x00\x00\x00\x05\x00\x00
\x00\x54\x00\x45\x00\x53\x00\x54\x00\x00\x00\xaa\xaa
\x05\x00\x00\x00\x00\x00\x00\x00\x05\x00\x00\x00\x54
\x00\x45\x00\x53\x00\x54\x00\x00\x00\xaa\xaa\x05\x00
\x00\x00\x00\x00\x00\x00\x05\x00\x00\x00\x54\x00\x45
\x00\x53\x00\x54\x00\x00\x00\xaa\xaa\x02\x00\x00\x00
\x44\x43\x42\x41\x02\x00\x00\x00\x00\x00\x00\x44

Supplementary Tools

- Some other tools we've developed
  - Automated IDL and IDB generation
  - PyDBG scripts for helping debug communication
  - Binary named pipe extraction
  - Symbol Inference

Automated IDL and IDB Generation

- Script recursively searches a directory
  - Looks for imports of RpcServer functions from rpcrt4.dll
- Launches IDA for each discovered module
  - Runs mIDA to pull out the IDLs and saves them
  - Saves the IDB for later perusal
  - All done in batch mode
- All done autonomously

PyDBG Scripts

- PyDBG is a scriptable python based win32 debugger
  - Part of the PaiMei RE Framework
- Script hooks key points in rpcrt4.dll
  - Dumps information about which NDR object is being parsed
  - Determines what caused bad stub data to be returned
  - Works on Windows XP SP2 and Windows 2000 SP4
- Works by looking up the format specifiers used internally to represent NDR types

Binary Named Pipe Extraction

- Simple IDC script for use in IDA
  - Extracts arguments to RpcServerUseProtseqEp
  - Provides the researcher with the named pipe name to use to connect to the server

Symbol Inference

- Some interfaces require proper contextual information for communication
  - User names, domain names, computer names, printers, shares, etc
  - Script attempts to automatically detect such requirements
  - For example, OpenPrinter in spoolss requires a valid printer name
- This allows for better code coverage and thus deeper auditings

Example Audits

- CA Brightstor ArcServe Backup
  - Going to walk through a quick audit

Example Audits

- Questions?
- For further questions, contact us via e-mail
  - Aaron Portnoy - aportnoy@tippingpoint.com
  - Cody Pierce - cpierce@tippingpoint.com

Total Slide Count

30