OPC Exposed – Part II

Ralph Langner
Langner Communications AG
Hamburg, Germany Lluis Mora
rl@langner.com

Abstract: It is well known that OPC does not include effective security controls and relies on DCOM. Well, the problem is much larger than that. In this paper, several DoS attacks that have proven effective against OPC servers are discussed that could be carried out by attackers with no technical background or by malware. In addition, a man-in-the-middle attack is explained that could be used by an aggressive attacker to have a SCADA system assume normal operation while the process is running wild. Last but not least, suggestions for remedies are presented.

Keywords: OPC, Denial Of Service, DCOM, Man-in-the-middle, Tunnelling, Vulnerability Testing, SCADA

1 Introduction

OPC architecture is somewhat typical for the industrial IT approach of the Nineties. Back then, "openness" was a good thing in itself and was (ab)used on any marketing occasion. Wouldn't it be great if all kinds of IT applications could interface process control equipment? "You bet!" – that's what most of us had thought back then. A more contemporary answer would be, "probably not".

OPC is the method of choice for standard Windows HMI and SCADA applications to interface automation peripherals. OPC interfaces have their well known security problems as they usually allow for manipulation of process variables without proper authorization. Anyway, this is not the focus of this paper. As a matter of fact, we don't discuss any (presumably unauthorized) write operation on OPC servers. After having seen several OPC servers in real production environments crash for no apparent reason, we did some investigation in OPC server stability and how OPC design features could be exploited either by malware or by an attacker. Most of the strategies discussed are low-level, brute force denial-of-service attacks. The problem with such DoS attacks is that they do not require an informed attacker, and may even be performed by malware, such as worms. The type of attacks highlighted here may be considered as a combined SCADA/IT attack, which we believe will become a prominent attack strategy of the future, as automation and IT is getting more and more interleaved.

For proof of concept, we implemented the attack methods described in a small Windows application, called OPC security checker. As the impact of each attack method depends on environmental factors (OS release, security patch, specific OPC server product), users of this software have the opportunity to check the consequences of an attack for their specific setup.

It is not our intention to criticize the OPC Foundation, Microsoft, or OPC server vendors for any possible design flaws that simply could not have been foreseen at design time. It is our intention to point out potential security issues in typical OPC deployments, to investigate their potential damage if exploited, and to point out why and where OPC users should care about OPC security more than what we see in everyday security assessments. The work presented here was not funded by any organization. It is part of our service for clients to help securing SCADA environments.

2 Typical OPC Environments, and How They Appeal to Attackers

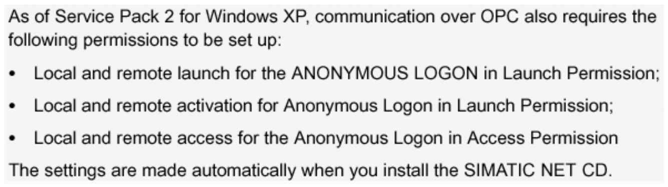
Most of the time when we take a closer look at OPC deployments in real life, we find old style, slow PCs with little memory and limited processing power, outdated operating system versions with no current security patches, improper DCOM access control, and staff with limited IT knowledge. All in all, pretty much appealing conditions to attackers and malware.

2.1 DCOM problem #1: Proper DCOM Security Settings Are Too Complex for Typical Staff and Typical Vendors

The OPC Foundation did their homework in the security department by defining the *OPC security interface*. Unfortunately, this interface is used on less than one percent of OPC deployments. The remaining 99+% rely on DCOM security settings for access control of standard DA servers. We have yet to see a company where DCOM access rights are configured properly. The reason is simple. The average process engineer who is in charge of configuring OPC servers is incapable of dealing with DCOM competently. If you take a closer look, it's easy to understand why.

DCOM security issues usually start with architecture. Proper authentication for DCOM objects can only be achieved if all participating PC's are part of a Windows domain. In this case, authentication is performed by LSASS against the Domain Controller. If the PC in question is not part of a Windows domain, as we often encounter in real life, authentication is performed by the local SAM (security accounts manager), which relies on userID plus password information.

DCOM configuration problems have exploded with every new operating system release and service pack. The current high water mark is constituted by XP SP2 or Windows 2003 SP1, where proper configuration of DCOM access rights for OPC servers requires a skilled IT technician. As vendors know that such a person is not always readily available, they advocate to <u>disable</u> DCOM security settings, or even do so automatically during installation. Quote from a leading vendor's manual:


As of Service Pack 2 for Windows XP, communication over OPC also requires the following permissions to be set up:

- Local and remote launch for the ANONYMOUS LOGON in Launch Permission;
- Local and remote activation for Anonymous Logon in Launch Permission;
- Local and remote access for the Anonymous Logon in Access Permission

The settings are made automatically when you install the SIMATIC NET CD.

Figure 1 – Vendor DCOM Advice

To make things worse, the installation procedure from the quoted vendor even resets your DCOM security settings when installing product updates.

2.2 DCOM problem #2: The Average OPC Installation is an Invitation to Attackers

The real security nightmare starts where DCOM access rights are reset not for a specific OPC server but for all DCOM objects, as suggested by several vendors.

Our experience is that the average process control person who installs an OPC server is not aware of the consequences of this approach. Therefore, we find many installations of OPC servers where DCOM access to the server box is wide open. This must look like an invitation to attackers. This given, plus the fact that we are dealing with outdated OS versions, plus the fact that many applications run with administrator rights (although not required), an attacker finds quite friendly conditions.

Anyone who resets *standard DCOM access rights*, as suggested by vendors, will probably not be aware of all the DCOM objects that are now accessible via the network. Just to name a few:

```
AccStore Class, Audiorecorder, Blocked Drivers, COM+ Event system,
ComEvents.ComServiceEvents, ComEvents.ComSystemAppEventData, Command
line Trigger Consumer, Event Object Change, Event Object Change 2,
FAT Defragmentation, HC, HTML Application, Internet Explorer (Ver
1.0), LegitCheckControl, MMC Application Class, Mobsync,
MSDAINITIALIZE, MSMQ, NetMeeting, NTFS Defragmentation, Paintbrush,
RDSessMgr, RDSHost, RemoteProxyFactory32 Class, Removable Storage
Manager, Removable Storage Sink Layer, Removable Storage UI Layer,
SENS Logon Events, SENS Logon2 Events, SENS Network Events, SENS
OnNow Events, SENS Subscriber for EventSystem, EventObjectChange
events, TimelinePreviewAX Class, TintSvr. upnpcont.exe, Volume Shadow
Copy Service, VssEvent, Watson subscriber for SENS Netword Events,
wiaacmgr, Windows Update Agent, zClientm
```

This is probably not what you would like to be accessible by anyone on your PC, as some of the objects mentioned have their own security flaws, such as buffer overflows, which might enable an attacker to gain administrative rights.

2.2.1 Architecture Problem: OPC Server Boxes Become Centralized

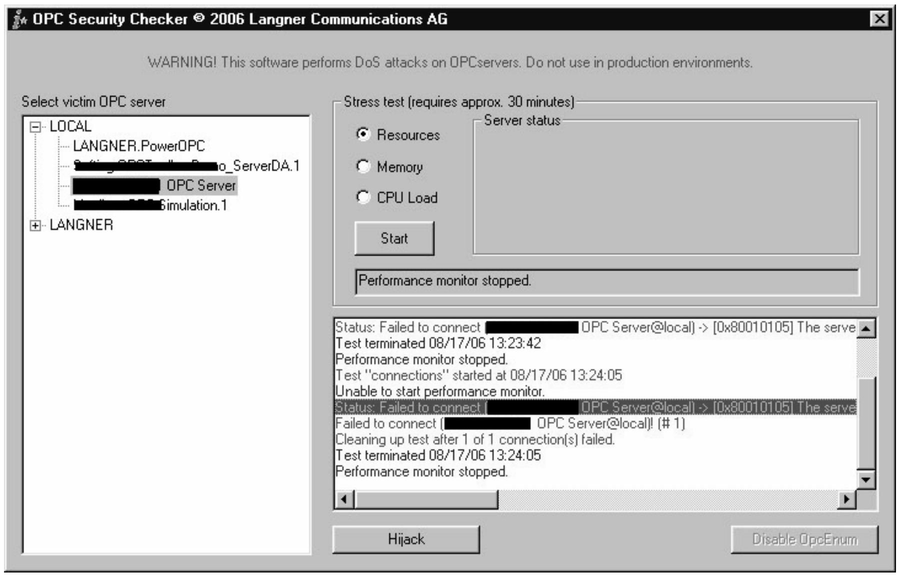
Back in the old days when automation peripherals were connected via point-to-point and fieldbus interfaces, OPC servers were deployed on remote PC's in close proximity to the peripheral, serving only a very limited number of peripherals. In more modern environments where automation peripherals are connected via Ethernet and TCP/IP, end users discover that they may well use one OPC server instance for a big number of identical peripherals. With Ethernet connectivity to peripherals, OPC server deployments get centralized. Such a box, connecting to hundreds of PLC's, becomes an interesting target for an attacker. If we manage to shut this one box down, operations of a complete factory may be affected.

3 Test Bench: The OPC Security Checker

In order to obtain realistic evidence on the impact that different attack strategies will have on given OPC environments, we implemented the attack methods discussed in this paper in a Windows application called the OPC Security Checker. By implementing and using this application, we were able to establish the exact steps for attack methods and to evaluate factors determining impact.

The OPC Security Checker application is also available for OPC server vendors, SCADA vendors and end users to evaluate potential impact of attacks in a lab scenario.

As a simple Windows application, this software just has to be launched on a client PC or on the server box itself.


OPC Security Checker © 2006 Langner Communications AG

WARNING! This software performs DoS attacks on OPCservers. Do not use in production environments.

Select victim OPC server

- LOCAL
  - LANGNER.PowerOPC
  - [redacted]o_ServerDA.1
  - [redacted] OPC Server
  - [redacted]Simulation.1
- LANGNER

Stress test (requires approx. 30 minutes)

(●) Resources
( ) Memory
( ) CPU Load

Server status

Performance monitor stopped.

```
Status: Failed to connect [redacted] OPC Server@local) -> [0x80010105] The serve
Test terminated 08/17/06 13:23:42
Performance monitor stopped.
Test "connections" started at 08/17/06 13:24:05
Unable to start performance monitor.
Status: Failed to connect ([redacted] OPC Server@local) -> [0x80010105] The serve
Failed to connect ([redacted] OPC Server@local)! (# 1)
Cleaning up test after 1 of 1 connection(s) failed.
Test terminated 08/17/06 13:24:05
Performance monitor stopped.
```

Figure 2 – OPC Security Checker

After the program starts, the desired victim OPC server is selected in the left tile. As can be seen in Figure 2, the target may either be local or remote. Selected target machines will be browsed for available OPC servers.

In the upper right tile, one out of three stress tests, which will be discussed below, may be selected. Once started, diagnostic information about the test in progress is displayed in the lower right tile. We also attempt to install a performance monitor on the target machine so that we are able to monitor system degradation without having to access the target machine's console.

The "Hijack" and "Disable OPCenum" buttons will execute the respective attacks as described below.

4 OPC servers under stress

The first set of attack methods is overloading an OPC server with too many and too heavy client requests, all inside the OPC interface specification.

4.1 Resources Exhaustion

The OPC interface definition does not specify an upper limit for OPC client connections. In typical production environments, we find between one to five OPC client applications that connect to an OPC server. What will happen with 100000+ client connections?

For the resources test, we simply keep connecting infinitely to the victim OPC server in a tight loop. The OPC server's `CoCreateInstanceEx` method is called to create a new instance of the `IID_IOPCServer` interface. No data is exchanged.

4.2 Memory Exhaustion

Any client that connects to an OPC server and is interested in process data must register an OPC group. An OPC group is kind of a shopping basket where the client application identifies all the items (ie. process variables) that it is interested in. Multiple OPC groups may be registered to request different update intervals for different types of data. Every OPC group is identified by a symbolic name that the client application may choose freely upon registration. What would a group name look like in real life? Probably "Temperatures", "Machine data", "Job data", or simply "a", "b", "c". The OPC interface specification does not place any constraints on group names.

For the memory test, we create one connection to the victim OPC server and keep creating OPC groups in a tight loop, using group names exceeding 10 megabytes. Even if it does not make any sense, this is specification conformant. Groups are added by calling the `AddGroup` method of the `IID_IOPCServer` interface. No data is exchanged.

4.3 CPU Overload

In this scenario, we attempt to bog the OPC server down by read requests. After having connected to the victim OPC server, we keep creating OPC groups in a loop, using "normal" (ie. short) group names. For every group, we use different update intervals to puzzle the OPC server's caching strategy. We put all items advertised by the OPC server into every group. Thereafter, we create a new client thread that performs a synchronous read (`Read` method of the `IID_IOPCSyncIO` interface) on all items.

5 Testing OPC Servers

For our tests, we configured one box as the victim machine and another box as the attacker. On the attacking system, a standard HMI application was installed to determine the effect of our attack on "normal" OPC client applications.

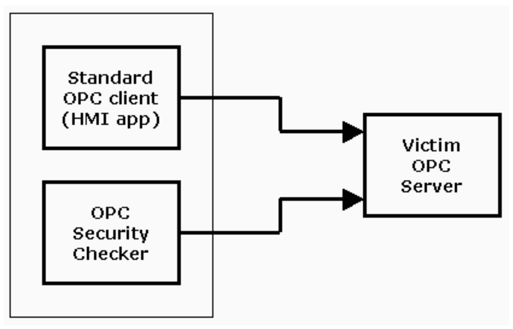


Figure 3 – Test Scenario

For the victim system we used three different boxes, equipped with Windows NT, Windows 2000, and Windows XP, all with latest service packs and patches installed (as of September 2006).

To determine the influence of specific OPC server implementations, we tested a total of eight different OPC server products (which we will not identify). Six OPC servers were connected to real PLCs, either by Modbus/TCP, by serial point-to-point interface (RS-232) or by some proprietary protocol over Ethernet. We also tested two simulation OPC servers with no connection to real peripherals.

5.1 Test Results

The testing saw a wide variety of dangerous results from uninterrupted operation to total system crashes. What we found is that the resulting behaviour is primarilhy dependent on the following parameters:

- operating system version
- specific OPC server implementation (vendor dependent)
- specific details of process connectivity

The biggest factor was found to be the operating system version.

> <u>Windows NT: Disastrous.</u> The most severe effects were encountered on a Windows NT box. One specific OPC server caused an out-of-memory situation during the memory test, which resulted in a complete system freeze. Most servers produced 100% CPU load for the resource test and CPU load test. Some OPC servers were unable to keep delivering process data, others only with significant delays.

Windows 2K: Unstable. With W2K, we did not see dramatic results as with NT. However, we did see 100% CPU loads and OPC servers that kept stopping and restarting continuously. One specific OPC server did discontinue delivering data with no error message.

Windows XP: Slow and stable, if you're lucky. XP showed the best performance. Only one of the OPC servers tested malfunctioned by creating new instances and having client applications reconnect manually. Again, expect 100% CPU loads and delayed data.

5.2 Impact of Performance Degradation

In general, it is hard to predict the impact of performance degradation on process control. Things to consider are:

- The degree of performance loss is dependent on the target machine's resources (CPU and memory) and operating system version.
- How tight is the client application's monitoring of process data? If the client application needs process data in real time, the impact will be much more significant than the impact for slowly changing information such as job data.
- How forgiving is the protocol used to access the peripheral(s)? With some old-style point-to-point protocols, inter-character delays of several hundred milliseconds will not be tolerated by the protocol, causing the peripheral to disconnect.
- What else is running on the target machine? Any critical IT application, such as a quality data logger, that happens to run on the victim box will be affected of performance degradation and might cause more ("collateral") damage than the attacked OPC server.

6 Bug or Feature?

Even though all observed behaviour probably is within OPC specification limits, we think that the following could be considered as software bugs:

- Spawning new OPC server instances. With most connectivity options to the peripheral, this will not yield any positive results, as connectivity resources are usually limited. You only have one TCP server port 502 on the other end, only one serial interface in your server PC, etc. As a consequence, the OPC servers tested were unable to access peripherals from their spawned instances.
- Lack of error messages for client applications. It looks like most OPC server vendors didn't take overload situations in account and have their server software remain silent in stress situations.

7 An All Too Easy Target: SCADA System with OPC Interface

Several vendors have equipped their SCADA systems with built-in OPC servers so that office, enterprise and MES applications may receive process information and aggregated production data from the SCADA system. While this sounds logical and straightforward, it comes at a price.

Remember that SCADA systems have problems when a virus scanner is launched on the same box. In average installations, the CPU load consumed by the virus scanner can cause the SCADA system to malfunction.

Now consider what will happen if we perform the stress tests discussed in this paper on an OPC server in a SCADA system. What we have seen is SCADA systems that freeze, even when running on Windows XP on current hardware. Part of the problem is that a SCADA system usually maintains a lot of process variables, five- or six-digit figures being common. This gives the attacker a lot to play with. If we apply the CPU overload attack, operation will continue smoothly only with a handful of OPC items configured. In real life settings, the SCADA system simply dies within seconds.

Unfortunately, some SCADA systems require DCOM for internal operation, so you can't just switch it off. The only solution in this case is to firewall DCOM ports.

7.1 Impact

If the SCADA system freezes, production will cease in most cases. This will be recognized by control room staff quickly. The question is how long the problem analysis and corrective action takes. In order to fix the problem, the attacking application must be identified and killed. This is probably out of scope for average control room staff. It might even be complicated if the attack is carried out from multiple stations (ie. DDos). We expect that in real life, it will take several hours until things are back to normal.

8 Denial of service, part two: Killing `OPCenum`

`OPCenum` is a Windows application that is usually installed along with OPC servers on the target machine. This software was developed by the OPC Foundation. It may be downloaded from the OPC Foundation's web site and is usually supplied with OPC server installation packages. `OPCenum` implements the `IOPCServerList` interface. It is used for two purposes:

- Determine the OPC servers installed on a specific target computer ("server browsing")
- Determine the CLSID of a specific OPC server, identified by its PROGID. A CLSID is required to actually connect to a COM object. If the CLSID is not specified in the OPC client software and only the PROGID is used, such clients will usually try to obtain the CLSID by calling *OPCenum* on the target machine.

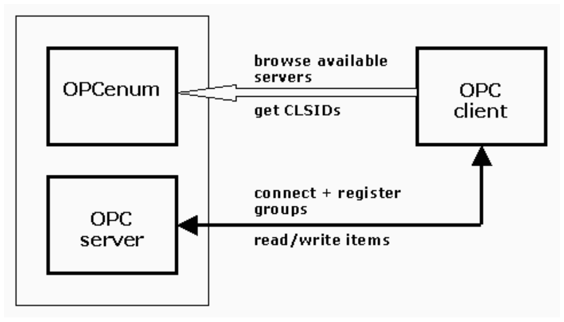


Figure 4 - OPCenum

If we disable *OPCenum* on the victim machine, server browsing will no longer be available. In addition, OPC client applications that utilize *OPCenum* to determine the target OPC server's CLSID will no longer be able to connect to their OPC servers.

To disable *OPCenum* on the victim machine, we use the Windows service control manager. In the service control manager, we set this service to SERVICE_DISABLED.

8.1 Required Access Privileges

- SERVICE_QUERY_CONFIG for locating *OPCenum*
- SERVICE_CHANGE_CONFIG for disabling *OPCenum*
- SERVICE_STOP for terminating *OPCenum*
- SERVICE_QUERY_STATUS for waiting until *OPCenum* actually stopped

8.2 Impact

In general, impact of this attack is low, as not all OPC clients are affected (only those that require server browsing, and those that use *OPCenum* for obtaining the OPC server's CLSID). If the attack succeeds, control room staff will usually determine quickly that a SCADA or HMI application is unable to connect to their data source.

Again, the interesting question is how long it takes to discover the real nature of the problem. While the fix is simple (restart *OPCenum*), it might take typical shift staff an hour or so to get things up and running again. If production has to be shut down in the meantime, damage might get significant. If the attacker has chosen to implement a looped attack routine (the procedure is carried out repeatedly so that the remedy is overwritten), especially skilled IT staff is required that might not be on shift. We would expect to see an attack like this not as a dedicated one-shot operation by a specific person but as a widespread malware attack.

9 Preparing for Maximum Damage: A Man-in-the-Middle Attack on OPC Connections

What we have learned from the 2003 power outage on the East Coast is that damage can increase tremendously if a process control system is unable to respond to alarms – or unable to detect alarm conditions. As most will remember, operators of the utility company recognized problems only when power backups in their control room kicked in, not by alarms from the SCADA system.

9.1 Architecture and Attack Stages

So what any sophisticated and aggressive attacker would try to achieve is to disable alarms of the control system before tampering with the process. The OPC architecture gives us a way to do this. Consider the normal OPC based SCADA system in Figure 5.

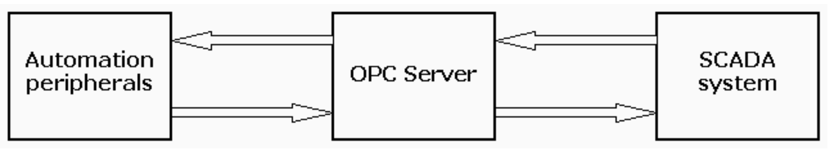


Figure 5 – Typical OPC in a SCADA System

As a first part of our attack, we insert a stealth OPC server between the original OPC server and the SCADA system as shown in Figure 6. We will explain later how this is done.

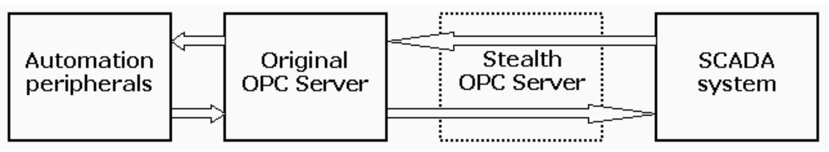


Figure 6 – Stealth Server Inserted in a SCADA System

The inserted stealth OPC server operates identically to the original OPC server. It exports the same items and values. Internally, for the original OPC server, the stealth OPC server appears as a client application. As data is passed back and forth unchanged, everything appears to be normal. Meanwhile, the stealth OPC server is learning normal operations for this setup.

When time has come to launch the real attack, the stealth OPC server switches from pass-through mode to simulation mode. The SCADA system now doesn't receive real process data, but simulated data for normal operations as shown in Figure 7.

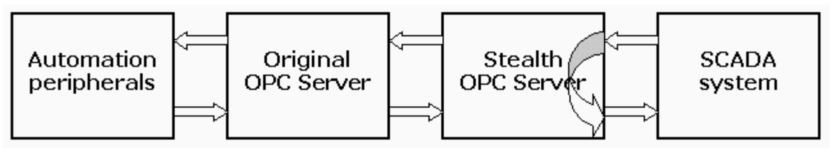


Figure 7 – Stealth OPC Server in Simulation Mode

The stealth OPC server may now manipulate the process, using the original OPC server, without having to worry that the SCADA system will detect any anomalies.

9.2 Procedure

1. Gain required access rights
2. Install the stealth OPC server on the victim machine
3. Rename the registry entry in LocalServer32 for the original OPC server to the path name of the stealth OPC server
4. Install a utility program on the victim machine for killing the victim OPC server process and execute the utility
5. The original OPC server process is killed, all client connections are lost
6. OPC client applications (including the SCADA system) will reconnect to the OPC server, however at this time the stealth OPC server is launched
7. Deinstall and delete the utility program to wipe out traces

9.3 OS Dependencies and Required Access Privileges

Hijacking OPC servers can be done on all Windows operating systems. Required access privileges:

- Remote registry access privileges
- KEY_ALL_ACCESS for the desired registry entry
- Privileges to access ADMIN$ on the victim machine
- SC_MANAGER_ALL_ACCESS and SERVICE_ALL_ACCESS for the Windows service control manager

You might argue that if an attacker has administrative privileges for the victim machine, he might do whatever he wants. While this is correct, the point is to illustrate how a sophisticated attacker would lay out an attack to produce maximum damage. Simply shutting down a SCADA system and erasing process archives would probably have less damaging effect than tampering with the process, especially with high energies and/or toxic substances, with all bells and whistles disabled.

9.4 OPC Hijacking in OPC Security Checker

In our test utility, we did not implement a full-blown stealth OPC server but went just half the way to demonstrate how the victim OPC server can be kicked out of business. We thought that a real implementation of a stealth OPC server is too dangerous at this time, with so many OPC installations being vulnerable against this type of attack.

What we did implement in the OPC security checker is the manipulation of the registry entry for the victim OPC server, plus killing all server instances. The registry entry is changed to an invalid path name by suffixing the original path name with ".hijacked". Once the victim OPC server instance is killed, OPC client applications are unable to reconnect, as the OPC server process can no longer be launched by Windows. As implemented, the OPC server hijack is another type of DoS attack.

After a successful hijack, the "Hijack" button of the OPC security checker changes to "De-Hijack". Pressing the button again will set things back to normal.

10 Lessons Learned and Recommendations

What we have found is that typical OPC deployments present various vulnerabilities. While discussions in the past highlighted the potential for unauthorized process manipulations, we have researched simple denial-of-service attacks. The problem with these attacks is that they may be carried out by people with zero technical background, even by malware. Many OPC deployments in real life are prone to such attacks. In most cases, the impact will either be loss of process control or loss/degradation of availability.

10.1 Threat Considerations

We believe that sooner or later denial-of-service attacks against OPC servers will be implemented in malware and distributed via the Internet as worms. Another means of distribution could be Trojan horses that look especially appealing to control room staff, maybe even disguising as neat OPC tools. The average malware author will probably (and hopefully) not include random writes to process variables. As a further means of disguise and to increase damage, such malware would presumably not execute an attack during launch but at some fixed point of time, like Sunday morning, 3 a.m., or December 25th.

We believe that a hijacking attack would typically be launched as a dedicated attack by terrorists or organized crime as part of a bigger plot against one specific target, such as a critical infrastructure facility.

In general, OPC vulnerabilities as shown offer a potential for combined SCADA/IT attacks. While DCOM and other IT security holes could be exploited by an attacker to take over control of a Windows PC running a SCADA system (what we have seen in the October 2006 Pennsylvania wastewater hack) simply to incorporate it in a Botnet, the point is that a malevolent, aggressive attacker would create more damage by focusing on the process control side of the equation.

10.2 Remediation Suggestions for OPC Server Vendors

We don't believe that DCOM security settings will be applied properly in the foreseeable future. However there are several things that could be done to improve security in existing environments. We suggest that OPC server vendors implement reasonable limitations into their products to keep the effects of a DoS attack low. Points to be considered:

- Limitations for client connections. There is no need for any OPC server to support 10000 or so client connections, so setting a reasonable limit is advisable.
- Limitations for OPC groups. There is no need for any OPC client to create 10000 or so groups.
- Limitations on group names. There is no reason for OPC group names to exceed 256 or so characters.
- Limitations on SyncIO operations. A client application that calls SyncIO in a tight loop may either be running wild, ill implemented, or trying to attack the OPC server.

10.3 Recommendations for End Users

- Think about using OPC tunnelling products. With OPC tunnelling, DCOM can be eliminated completely (by firewall) for access to an OPC server.
- Configure DCOM access rights properly. There is no excuse for leaving DCOM access wide open, and vendors should provide appropriate tutorials along with their products to illustrate the necessary steps to be taken.
- Update your operating system. Tests results have shown that DoS attacks are much less dramatic on XP than on other (older) platforms.
- OPC access to SCADA systems must be handled with extra care. A VPN tunnel to the SCADA system's OPC server should be considered.

10.4 Recommendations for SCADA System Vendors and System Integrators

- Implement extensive diagnostic information about OPC connections.
- Think twice about using an OPC server integrated in a SCADA system. This makes a SCADA installation very vulnerable.
- Educate your end users in evaluating security risks, and especially in DCOM and firewall issues.
- Perform DCOM security settings properly.

About the Author – Ralph Langner has accumulated 19 years experience in industrial IT. He started his company implementing low level protocol driver software. Now known as Langner Communications AG, Ralph's company offers a wide range of component software and middleware for connecting to automation peripherals, from embedded PLC drivers up to EAI connectors. Several years ago, Langner Communications determined that with all the ongoing networking of automation peripherals, SCADA systems and IT applications, most people had simply forgotten about implementing proper security boundaries and procedures. Industrial IT security has become one of Langner Communications' main focuses. For more information on Langner Communications, please refer to www.langner.com.