VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

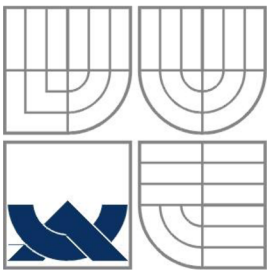
PODPORA MS-NAP PRO SYSTÉM AVG

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

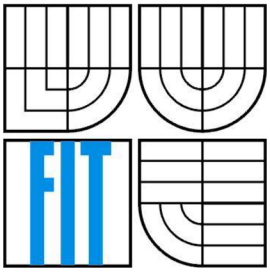
AUTOR PRÁCE
AUTHOR

ONDŘEJ KUCHTA

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PODPORA MS-NAP PRO SYSTÉM AVG

MS-NAP SUPPORT FOR AVG SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ONDŘEJ KUCHTA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PAVEL OČENÁŠEK, Ph.D.

ZDE VLOZTE ZADANI

BRNO 2011

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací podpory technologie NAP společnosti Microsoft do antivirového produktu AVG společnosti AVG Technologies.
Začátek práce obecně popisuje architekturu technologie NAP a možností jejího nasazení. Dále se práce zabývá návrhem a implementací klientské i serverové části podpory NAP pro antivir společnosti AVG. Výsledkem je ucelená práce pojednávající o implementaci a nasazení NAP do produktů třetích stran.

Abstract

This bachelor thesis deals with design and implementation of Microsoft NAP technology for AVG antivirus from the AVG Technologies Company. First part describes general architecture and deployment options of NAP technology. The next part describes design and implementation of client-side and server-side NAP support for AVG antivirus. The result is a comprehensive document about design and implementation of NAP for third-parties software solutions.

Klíčová slova

NAP, AVG, SHA, SHV ,SoH, SoHR .

Keywords

NAP, AVG, SHA, SHV ,SoH, SoHR .

Citace

Ondřej Kuchta: Podpora MS-NAP pro systém AVG, bakalářská práce, Brno, FIT VUT v Brně, 2011

Podpora MS-NAP pro systém AVG

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Pavla Otčenáška, Ph.D. Další informace mi poskytli především zaměstnanci společnosti AVG Technologies.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Ondřej Kuchta
Datum 17.5 2011

Poděkování

Rád bych poděkoval především Eriku Pomykalovi za jeho trpělivost a ochotu. Všem členům AVG Enterprise teamu, kteří mi poskytli obrovské množství potřebných informací a celé společnosti AVG Technologies CZ, s.r.o. za poskytnutí technického zázemí pro tvorbu této práce.

Obsah

1 Úvod

Prosazování a dodržování bezpečnostních politik pro připojení především vzdálených klientů k podnikovým sítím je v dnešní době velice obtížným a stěžejním úkolem. Je to dáno nejen zvyšujícím se počtem škodlivého software, ale především využívání mobilních zařízení pro připojení k podnikovým sítím, jejichž počet se každým rokem nezanedbatelně zvyšuje. Administrátor ztrácí přehled o stavu zařízení připojovaných k síti, protože mobilní zařízení obvykle nespadají pod administrátorovu správu a nemá tedy přehled, zda je každé zařízení v síti chráněno antivirem, firewallem a dalšími bezpečnostními prvky a zda není infikováno škodlivým software. Především z těchto důvodů stále vznikají technologie, které umožňují hlídat stav klienta a prosazovat bezpečnostní politiku sítě. Technologie NAP (Network Access Protection) vyvinutá společností Microsoft [5] je jeden z klíčových nástrojů na prosazování bezpečnostních politik pro přístup k síťovým zdrojům.

Nasazení technologie NAP spočívá v implementaci podpory NAP do bezpečnostních produktů třetích stran. Účelem této práce je navrhnout a implementovat rozšíření Antiviru od společnosti AVG o podporu NAP. Vedlejším výstupem práce je obecný návod na návrh a implementaci podpory NAP do libovolného produktu třetích stran.

1.1 Cíle

Obecně popsat architekturu technologie NAP a způsob jejího nasazení. Navrhnout a implementovat serverovou i klientskou část podpory NAP pro antivir společnosti AVG. Cílem je rovněž ucelená práce pojednávající o implementaci NAP a nasazení NAP s produkty třetích stran.

Antivir společnosti AVG bude schopen odesílat informace o stavu počítače, stáří virové databáze, nastavení firewallu, nastavení komponent, informace o infekcích. Rovněž bude schopen provádět skenování klienta před připojením, spouštět a nastavovat komponenty tak, aby vyhovovaly požadavkům sítě a odstranit všechny případné infekce dřív, než bude klient do sítě připuštěn.

1.2 Pojem NAP

NAP je nová platforma vyvinutá společností Microsoft, která kontroluje přístup k síťovým zdrojům na základě identity a dodržení bezpečnostní politiky sítě klientského počítače.[5]

S produktem podporujícím technologií a protokol NAP je administrátor podnikové sítě schopen definovat minimální požadavky na stav každého připojovaného klienta. Na základě informací o stavu přijatých od klienta může server podle pravidel vyhodnotit, zda bude klient připuštěn do sítě či nikoliv. Technologie NAP neslouží k zabezpečení sítě proti vniknutí, ale pouze jako doplněk pro dodržování bezpečnostních zásad a pravidel pro přístup především k podnikovým sítím[1], kde je důležité, aby každý klient dodržoval daná bezpečnostní pravidla.

Pokud klient nesplňuje bezpečnostní politiku sítě, NAP poskytuje mechanismy, které klienta do sítě nepřipustí a zároveň automaticky napraví stav klienta tak, aby byl akceptován při dalším pokusu o připojení. Když se klient připojí k síti, zajišťuje architektura NAP trvalou kompatibilitu jeho stavu.

K použití architektury NAP dojde vždy, když se klient pokusí připojit k síti prostřednictvím serveru pro přístup k síti, například server virtuální privátní sítě (VPN), nebo pokud se klient pokusí navázat komunikaci s jinými síťovými prostředky[2].

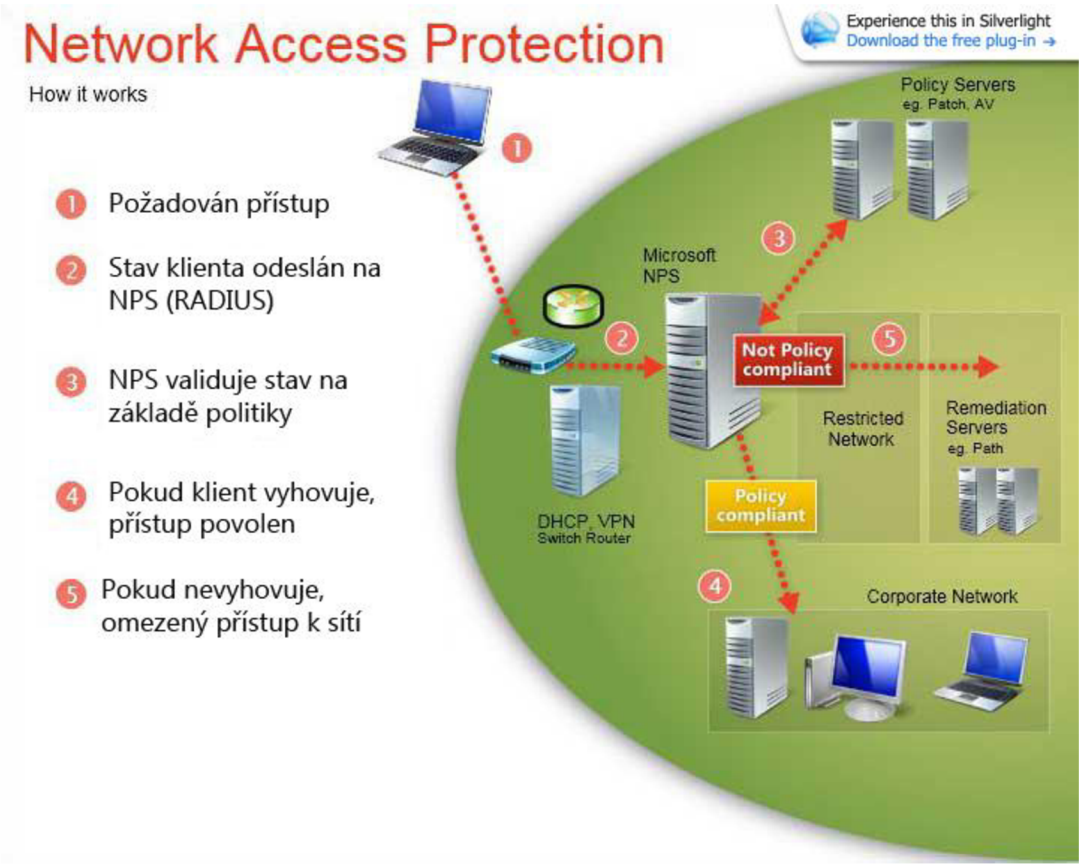
1.3 Kdy používat MS-NAP

Technologie NAP je použitelná v sítích postavených na řešeních společnosti Microsoft.[5] Základním požadavkem je Microsoft server 2008 a vyšší na straně serveru a operační systém Windows XP SP3 a vyšší na straně klienta[6]. Aby byl systém plně použitelný, je potřeba zajistit, aby klienti používali bezpečnostní software, který technologii NAP podporuje. Základními scénáře pro využití NAP mohou být například[8]:

- Vynucení kontroly stavu cestovních přenosných počítačů při jejich znovu připojení k podnikové síti.
- Ověření odpovídajícího stavu u domácích počítačů, které se připojují k podnikové síti pomocí serveru VPN.
- Ověření stavu a omezení přístupu přenosných počítačů přinesených do organizace návštěvníky.

1.4 Síťová architektura NAP

Tato kapitola vychází z [7]



1.4.1 Síťová architektura NAP[1]

Základní komponenty síťové architektury NAP [8] :

- **Klient.** Počítač s operačním systémem Windows XP SP3 a vyšším, který využívá software s podporou NAP a žádá o přístup do sítě.
- **NAP ES (Enforcement Server) -** Server vynucení**.** Tento server kontroluje přístup k síti, dokud klienti nepotvrdí svůj kompatibilní stav splňující bezpečnostní pravidla sítě. Pak umožní klientovi buď připojení přímo k síti, nebo klienta připojí pouze k oblasti s omezeným přístupem.
- **Server NPS (Network Policy Server) -** Server běžící pod operačním systémem Windows Server® 2008, který vynucuje připojení klientů k síti podle vytvořených zásad, týkajících se především stavu klienta a autorizace připojení.
- **Policy Servers -** Servery politik.

  Tyto servery definují aktuální kompatibilní stav. Například server, který zjišťuje informace o aktuální verzi antivirového software AVG, je dotazován na tuto informaci serverem NPS při

[1] Obrázek vychází z obrázku: [cit. 2011-05-11]
<http://www.microsoft.com/global/windowsserver2008/en/us/PublishingImages/NAPFeatAlt.jpg>

každém připojení klienta. Tímto je dosaženo toho, že klient se starou virovou definicí nebude připuštěn do sítě, dokud neaktualizuje svojí virovou databázi.

- **Restricted network -** Oblast s omezeným přístupem.
    - **Remediation servers -** Servery použitelné pro uzdravení klienta.
      Servery hostující softwarové updaty a nejnovější antivirové databáze, signatury.
    - **Klienti s omezeným přístupem**
      Klienti, kteří ještě nesplňují požadavky sítě.
    - **Klienti bez podpory NAP**
      Klienti, kteří nepodporují technologii NAP a pravděpodobně nebudou. Jedná se především o mobilní zařízení, jakou jsou mobilní telefony a podobně. Je možné těmto zařízením například udělit výjimku pro přístup do sítě, což se ovšem nedoporučuje.

Pro NAP ES servery vynucení existují čtyři **NAP vynucovací metody**[7] dodávané přímo společností Microsoft, které omezují klientovo připojení k síti, dokud klient neprokáže, že splňuje požadavky sítě. Tyto metody je možno kombinovat pro větší robustnost.

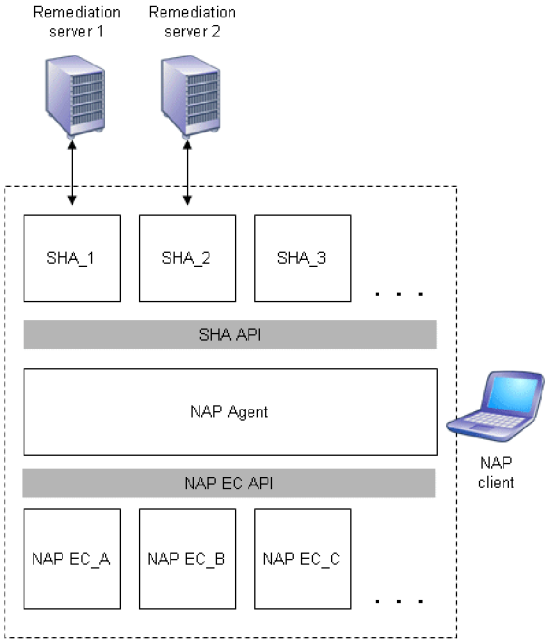
- **Vynucení IPsec.** Pokud je použita metoda vynucení IPsec, může klient komunikovat jen s omezeným počtem serverů, dokud neprokáže, že jeho stav odpovídá požadavkům sítě.
- **Vynucení 802.1X.** Metoda používána jak pro drátové, tak bezdrátové sítě. Omezení je vynuceno pomocí ACL listu, popřípadě může být klient umístěn do vyhrazené VLAN sítě.
- **Vynucení VPN**. Pokud je použita tato metoda vynucení, tak sám VPN server zakáže přístup nekompatibilního klienta pomocí IP filtru, dokud klient neprokáže svůj stav.
- **Vynucení DHCP**. Server DHCP přidělí klientovi, který neprokázal svůj stav, nebo se jeho stav neshoduje s požadavky sítě, IP adresu verze 4 se kterou se může pohybovat pouze v rámci sítě s omezeným přístupem. Pokud se klient prokáže a jeho stav bude odpovídat požadavkům sítě, tak server DHCP pošle nové informace a přidělí klientovi novou IP adresu. Tento typ vynucení se moc nedoporučuje i přes jeho jednoduchost.

Popis možností a použití a sestavení sítí je nad rámec této publikace. Existuje obsáhlý volně stažitelný dokument vydaný společností Microsoft popisující správu sítí s podporou technologie NAP. [6]

2 Architektura NAP

2.1 NAP architektura klienta

Tato kapitola vychází ze zdroje[7].



2.1.1 NAP architektura klienta[2]

NAP EC (Enforcement Client) klient vynucení**.**

Pro každý zavedený přístup k síti a komunikaci je definován zvláštní klient vynucení. Klienti vynucení pro IPsec, 802.1X, DHCP a VPN jsou již dodáváni s platformou NAP. Rovněž je možné využít rozhraní NAP EC API a vytvořit vlastního vhodného klienta vynucení. NAP EC klient předává informace o stavu vzdálenému serveru, kde běží odpovídající NAP ES. Získává informace o stavu od NAP agenta.

NAP agent

Je již existující a běžící služba v rámci operačního systému klienta, který podporuje technologii NAP. Jejím hlavním úkolem je sbírat zprávy SoH od všech SHA zaregistrovaných v systému a tyto informace poskytovat NAP EC, které je odešle na vzdálený server. Data získané z SHA jsou dále porovnávána pomocí SHV (System Health Validator) umístěného na vzdálené straně serveru, který na základě těchto dat vyhodnotí stav klienta jako vyhovující nebo nevyhovující. Rovněž přímá data od serveru a předává zpět ve formě SoHR zprávy konkrétnímu SHA.

[2] Zdroj [7] s 7.

Hlavní činnosti NAP agenta[7]:

- Sbírá zprávy SoH od všech SHA klientů v systému a ukládá je.
- Poskytuje NAP EC klientovi na vyžádání uložené zprávy SoH.
- SoH je aktualizováno, když některý z SHA nahlásí změnu stavu produktu. Například pokud se změní stav libovolné komponenty antiviru AVG.
- Informuje SHA pokaždé, když se změní stav připojení klienta ke vzdálené síti.
- Poskytuje SoHR odpovídajícímu SHA.

SHA (System Health Agent)

Agent zdraví je komponenta vždy vytvořená třetí stranou, která je schopna komunikovat s konkrétním produktem. Je hlavním předmětem návrhu a implementace klientské strany. Provádí update klienta a informuje ve formě SoH zprávy agenta NAP o stavu klienta. Zároveň i přímá odpovědi od NAP agenta ve formě SoHR zpráv obsahující informace, zda byl stav klienta vyhodnocen jako vyhovující, popřípadě nevyhovující a především z jakých důvodů. SHA agent je schopen v případě nevyhovujícího stavu na základě přijatých informací upravit nastavení antiviru AVG tak, aby byl klient při příštím pokusu o připojení shledán vyhovujícím. Komunikuje s NAP agentem pomoci SHA API vrstvy. Agent SHA musí být v systému zaregistrován, aby byl schopen komunikace s NAP agentem.

Zpráva SoH (Statement of Health)

Je zpráva předávaná mezi SHA a NAP agentem. Jedná se o strukturu obsahující informace o aktuálním stavu klienta, která je dále odesílána NAP agentem ke klientovi vynuceni NAP EC. Klient vynucení odešle SoH ke zpracování na vzdálený server.

Zpráva SoHR (Statement of Health Respond)

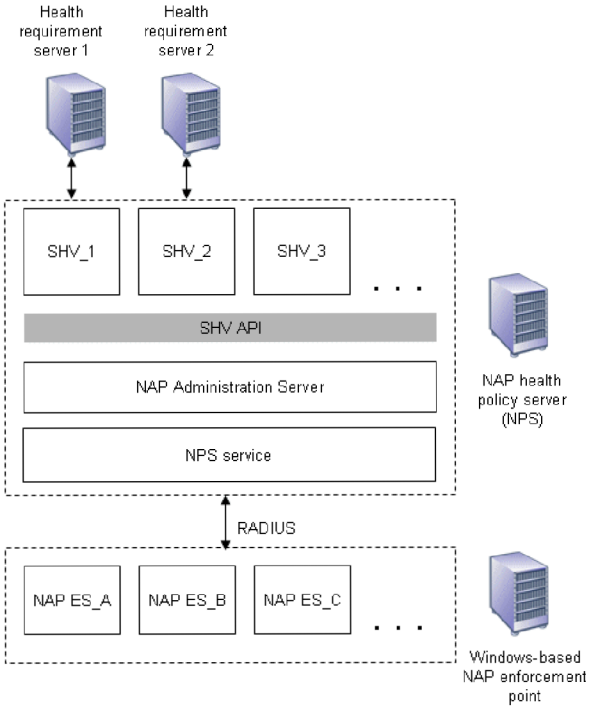
Je zpráva předávaná mezi SHA a NAP agentem. Jedná se o strukturu obsahující odpověď od serveru o stavu klienta, kterou NAP agent přijal od NAP EC a předá vždy konkrétnímu SHA, kterému informace patří. SoHR obsahuje obvykle i informace o důvodu odmítnutí klienta, které následně agent SHA využije k jeho nápravě.

SHA API

Poskytuje programové rozhraní, které umožňuje přístup ke službě NAP Agent, komunikaci se službou NAP, odesílání stavu klienta ve formě SoH zprávy a příjmat SoHR zprávy. Rozhraní je použito při tvorbě SHA agenta pro antivirus AVG. Je podrobně probráno v kapitolách návrhu a implementace.

2.2 NAP architektura serveru

Tato kapitola vychází ze zdroje[7].



2.2.1 NAP architektura serveru[3]

NAP ES (Enforcement Server) Server vynucení.

Pro každý zavedený přístup k síti a komunikaci je definován zvláštní klient vynucení. Server vynucení pro IPsec, 802.1X, DHCP a VPN je již dodáván s platformou NAP. Rovněž je možné využít rozhraní NAP EC API a vytvořit vlastní vhodný server vynucení. NAP ES přímá zprávy SoH od odpovídajících NAP EC na klientské straně a předává je NAP health policy serveru. Pokud tedy běží na serveru NAP ES pro přístup pomocí DHCP, tak je odpovídajícím NAP EC na klientské straně NAP EC pro DHCP přístup.

NAP ES dodávány s Windows Server 2008 jsou [5]:

- IPsec NAP ES pro IPsec zabezpečenou komunikaci
- DHCP NAP ES pro konfiguraci síťových adres založenou na DHCP.
- 802.1X NAP ES
- VPN NAP ES

[3] Zdroj [7] s 11.

NAP Administrační Server.

NAP administrační server provádí tyto činnosti:

- Získává zprávy SSoH od NAP ES přes NPS službu.
- Distribuuje SoH vždy konkrétnímu SHV.
- Sbírá zprávy SoHR od SHV a předává je NPS službě.

NPS služba

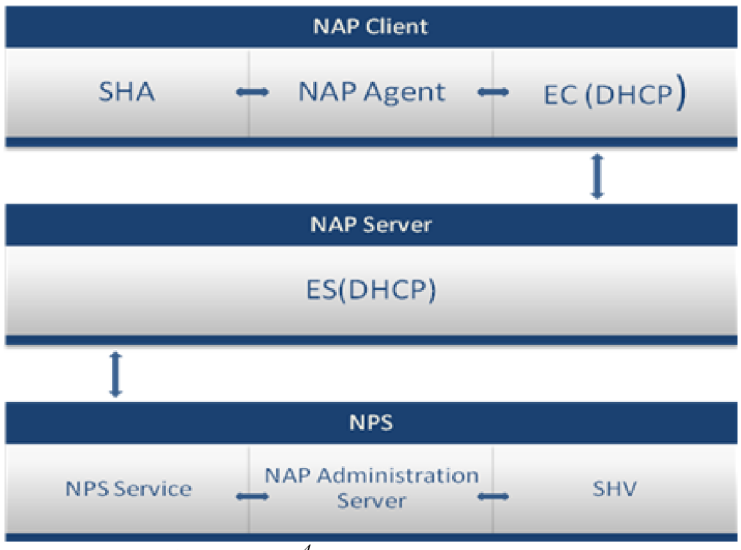
NPS služba komunikuje s NAP serverem pomocí protokolu RADIUS. Je to rozhraní mezi NAP serverem a NAP administračním serverem. Na základě SoHR od SHV vytváří služba NPS zprávu SSoHR, která stanovuje, zda je klient vyhovující či nikoliv a přidává do této zprávy rovněž zprávu SoHR od komponenty SHV.

SHV – (System Health Validator)

Je komponenta vždy vytvořená třetí stranou, která běží na straně serveru. SHV přímá zprávy SoH od NAP administračního serveru a porovnává informace o stavu klienta, které jsou ve zprávě obsaženy, se stavem který je vyžadován politikou sítě.

SHV API

Poskytuje programové rozhraní, které umožňuje přístup k NAP Administračnímu serveru, komunikaci, příjem SoH zpráv o stavu klienta a odesílání SoHR zpráv o výsledku validace a chybách na straně klienta. Rozhraní je použito při tvorbě SHV validátoru pro antivirus AVG. Je podrobně probráno v kapitolách návrhu a implementace.



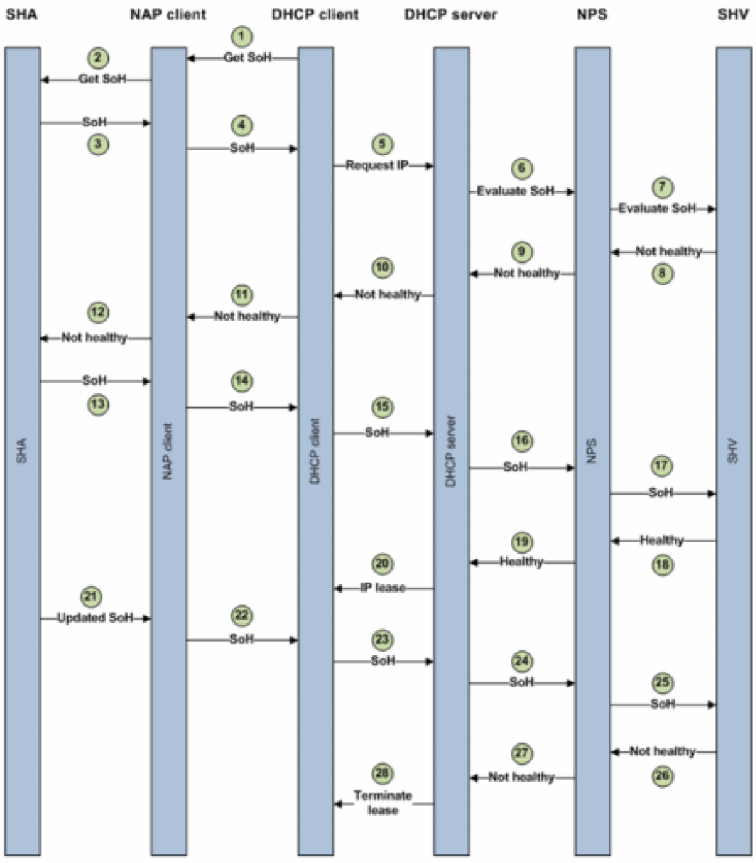
2.2.2 Infrastruktura NAP.[4]

[4] Zdroj: [cit. 2011-05-11] <http://4sysops.com/wp-content/uploads/2007/09/nap.png>

2.3 Komunikace mezi klientem a serverem

Tato kapitola vychází ze zdroje[7].

Jak již bylo zmíněno výše, systém NAP komunikuje pomocí zpráv SoH a SoHR. Scénář komunikace mezi NAP klientem a NAP serverem.



2.3.1 Komunikace mezi klientem NAP a serverem NAP.[5]

- Klientská stanice očekává IP adresu od DHCP serveru. DHCP klient je ovšem nastavený na NAP a výsledkem je, že DHCP klient kontaktuje NAP klienta a vyžaduje od něj zprávy SoH.
- NAP klient si vyžádá všechny zprávy SoH od všech SHA agentů v systému.
- NAP klient předá kolekci zpráv SoH klientu DHCP.
- DHCP klient pošle dotaz na IP adresu serveru DHCP. Součástí dotazu je i kolekce SoH.
- DHCP server musí předat SoH k validaci serveru NPS, který rozhodne, zda je klient vyhovující.
- Pro každého SHA agenta běžícího na klientské stanici musí existovat validátor SHV na serveru NPS, který je schopen validovat data od určitého SHA agenta.

[5] Zdroj: [cit. 2011-05-11] <http://i.msdn.microsoft.com/dynimg/IC87598.gif>

- V tomto scénáři byl z názorných důvodů klient shledán nevyhovujícím, protože nesplňoval bezpečnostní politiku sítě nastavenou na NPS. To znamená, že některý z SHA agentů odeslal data, která byla vyhodnocena jako nevyhovující, protože komponenta, kterou agent SHA hlídá, není ve stavu, ve kterém by podmínky sítě splňovala. SHV může v tomto případě odeslat zpět informace, proč nebyl klient shledán vyhovujícím, které agent může použít k automatické nápravě stavu komponenty.
- Zpráva SoHR s odpovědí od serveru projde zpět přes všechny prvky systému NAP.
- Pokud navrácená zpráva obsahuje informaci o tom, že klient nebyl shledán jako vyhovující, tak může konkrétní SHA agent začít pracovat na automatické opravě komponent a pokusit se o připojení znovu.

Komunikace mezi komponentami klienta a serveru. Vychází z [6]

Komunikace mezi NAP agentem a NAP administračním serverem:

- NAP Agent předá SSoH k NAP EC.
- NAP EC předá SSoH k NAP ES.
- NAP ES předá SSoH k NPS službě.
- NPS služba předá SSoH k NAP administračnímu serveru.

Komunikace mezi NAP administračním serverem a NAP agentem:

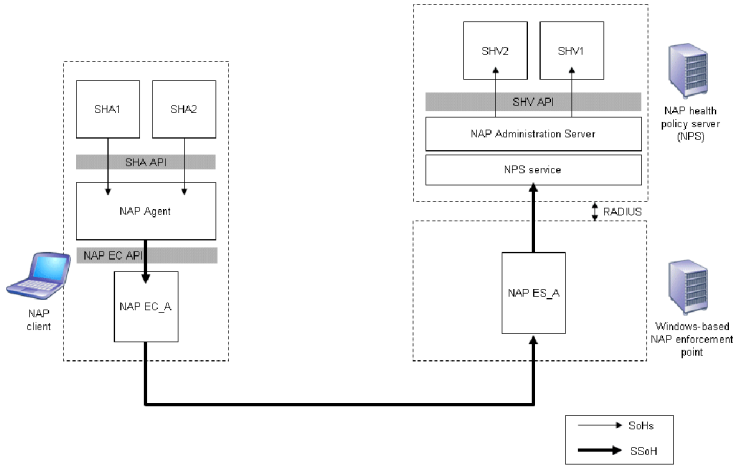
- NAP administrační server předá SoHR k NPS službě.
- NPS služba předá SSoHR k NAP ES.
- NAP ES předá SSoHR k NAP EC.
- NAP EC předá SoHR k NAP Agentovi.

Komunikace mezi SHA agentem a SHV:

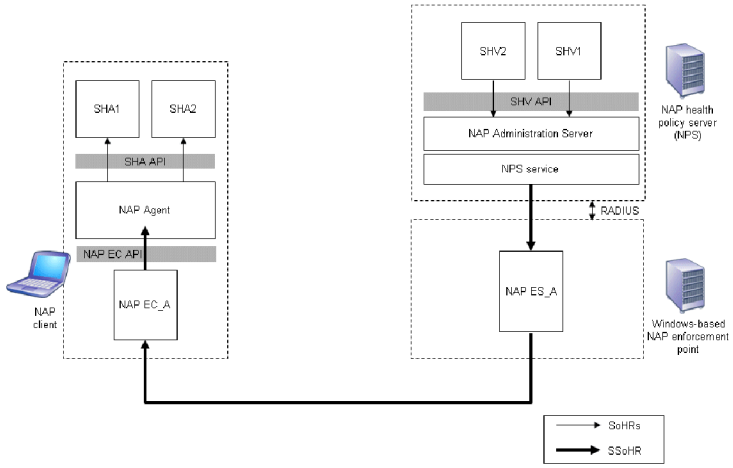
- SHA předá svoji SoH zprávu NAP Agentovi.
- NAP Agent předá SoH, zabalenou do SSoH, klientu vynucení NAP EC.
- NAP EC předá SSoH ke vzdálenému NAP ES.
- NAP ES předá SoH k NAP administračnímu serveru.
- NAP administrační server předá SoH k SHV.

Komunikace mezi SHV a SHA agentem:

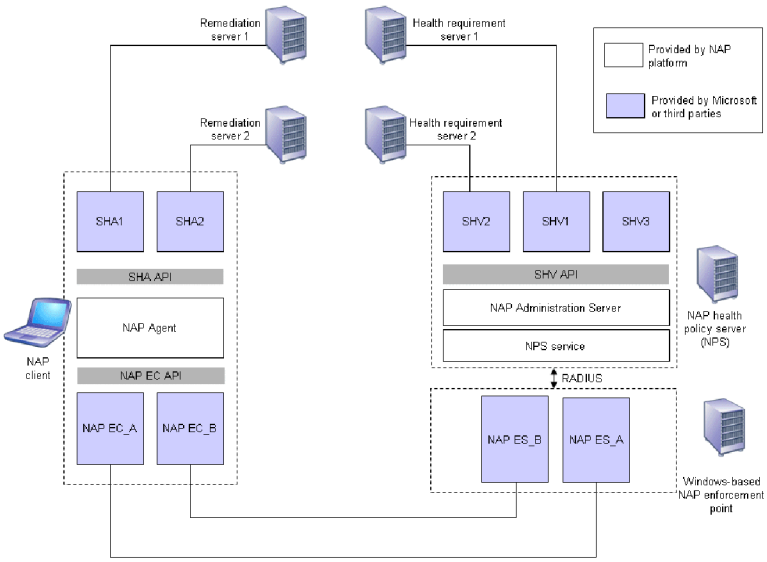
- SHV předá odpověď SoHR k NAP administračnímu serveru.
- NAP administrační server předá SoHR ke NPS službě.
- NPS služba předá zprávu SoHR, zabalenou do SSoHR, k NAP ES.
- NAP ES předá SSoHR k NAP EC.
- NAP EC předá SoHR k NAP agentovi.
- NAP Agent předá SoHR k SHA.



2.2.2 Komunikace mezi komponentami NAP klienta a NAP serveru[6]



2.2.3 Komunikace mezi komponentami NAP serveru a NAP klienta[7]



2.2.4 Vztah mezi komponentami NAP klienta a NAP serveru.[8]

[6] Zdroj [7] s 16.
[7] Zdroj [7] s 17.
[8] Zdroj [7] s13.

3 Analýza, návrh a implementace komponent klienta architektury NAP

Cílem je navrhnout komponentu klienta NAP, která bude mít za úkol sbírat informace o stavu počítače z antiviru AVG a tyto informace poskytovat službě NAP na vzdáleném serveru. Kdykoli se změní stav některé z kontrolovaných komponent antiviru AVG, zaslat serveru nový stav na validaci. Pokud není nový stav v souladu s politikou serveru, bude klient od sítě odpojen. Komponenta musí být rovněž schopna přímá informace ze serveru a zajistit nápravu stavu do stavu požadovaného. To znamená, provést aktualizaci antiviru AVG na straně klienta, spustit některou z deaktivovaných komponent (např. Resident-Shield) a rovněž informovat uživatele o nesouladu stavu s politikou podnikového serveru a nemožnosti se připojit. Tyto základní požadavky vychází jednak z architektury a možností použití technologie NAP[5] a rovněž z požadavků společnosti AVG.

3.1 Návrh agenta SHA

Základní požadavky na agenta SHA vychází z jeho obecně dané architektury dokumentované společností Microsoft a způsobu nasazení v operačním systému na straně klienta. Seznam požadavků je dále rozšířen o požadavky společnosti AVG navazující na funkcionalitu agenta.
Základní požadavky na agenta SHA jsou:

- Musí v systému běžet jako služba.
- Připojení pomocí COM ke službě NAP agent.
- Musí implementovat všechny dané callback funkce architektury NAP.
- Komunikace se službou NAP agenta.

Rozšířené požadavky na agenta SHA:

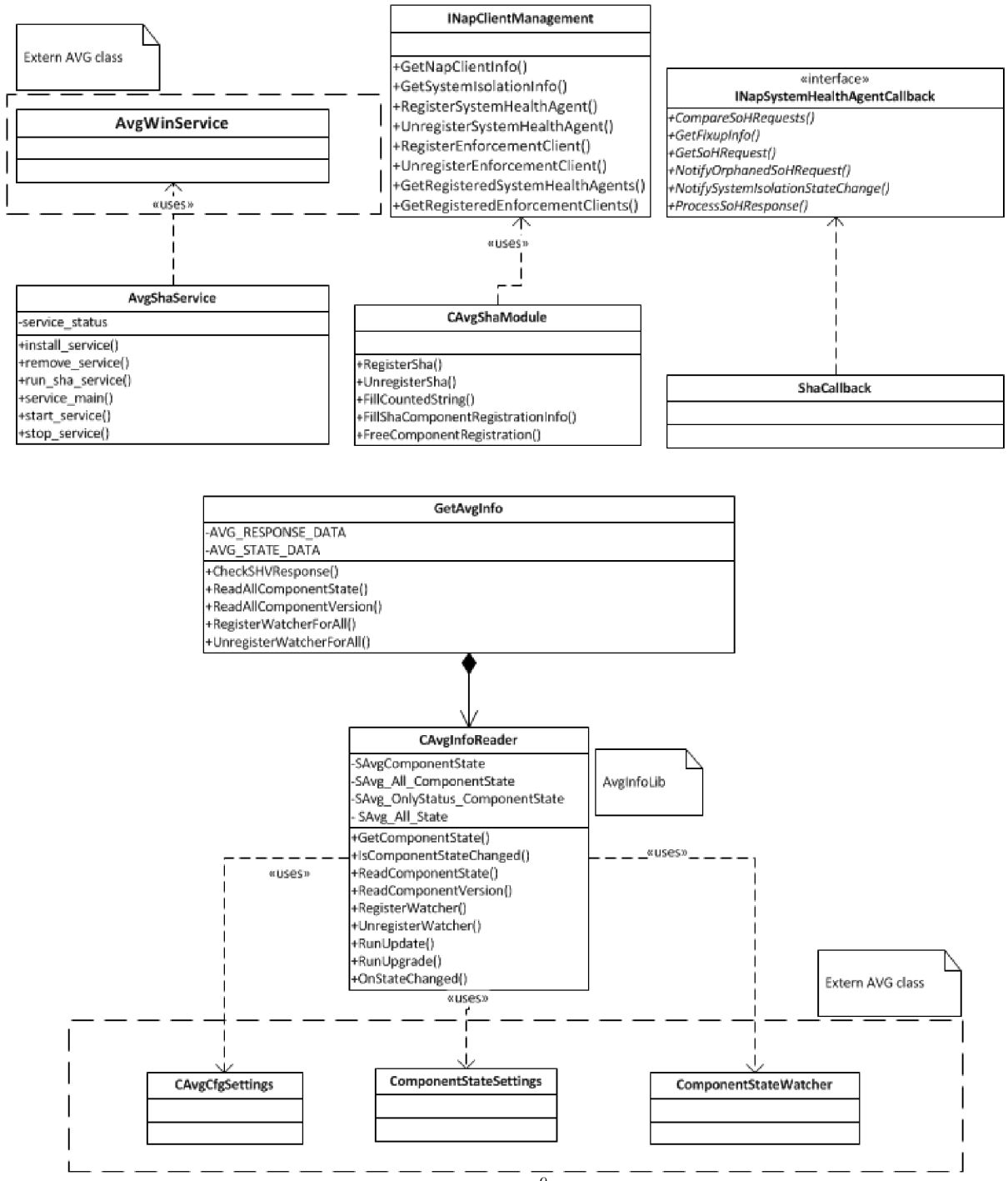
- Sběr dat z antiviru AVG.
- Kontrola změny stavu komponent antiviru AVG.
- Nastavit antivir AVG do stavu akceptovatelného politikou serveru.
    - Spouštění a vypínání komponent antiviru včetně spouštění skenování.
    - Nastavení komponent antiviru včetně nastavení firewallu.
    - Spouštění update a upgrade antiviru a jeho virové databáze.
- Možnost úprav agenta v případě rozšíření počtu komponent antiviru.

Aplikace je rozdělena na dvě části. Na samotného SHA agenta, který bude komunikovat s NAP agentem a rovněž na knihovnu, která bude obstarávat abstrakci nad komunikací s produktem AVG. Je to především z důvodů možnosti pozdější rozšiřitelnosti agenta SHA, popřípadě použitelnosti agenta s jiným produktem.

Protože komponenta musí běžet jako služba v systému windows[7], tak bude existovat třída **AvgShaService**, jejíž metody se postarají o registraci komponenty jako služby a zároveň o zavedení komponenty do systému. Třída využívá instance externí třídy **AvgWinService,** proto mezi nimi platí závislost „uses“. [4]
Dalším požadavkem na SHA agenta je jeho zaregistrování v systému, aby byl schopen komunikace s NAP agentem[5].Pro registraci SHA v systému slouží třída **CAvgShaModule**, která má metody **RegisterSha** a **UnregisterSha** pro případné odstranění SHA agenta ze systému.
NAP agent volá metody třídy **ShaCallback** jejíž deklarace je dána rozhraním **INapSystemHealthAgentCallback** a musí ji implementovat každy SHA agent v systému. ShaCallback je implementací tohoto rozhraní. Metody z třídy ShaCallback jsou volány pokaždé ,když NAP agent vyžaduje data o stavu produktu, který SHA agent hlídá.[7]
Poslední třídou je **GetAvgInfo**, která je navržena tak,aby sloužila jako rozhraní mezi knihovnou AvgInfoLib popsanou dále a SHA agentem. Slouží pro získávání informací o stavu antiviru AVG a pro nastavení jeho komponent.

Další součástí SHA agenta je knihovna **AvgInfoLib**, která komunikuje s antivirem AVG a získává potřebné informace o jeho stavu. Zároveň má přístup k jeho nastavení a ovládání. Třída **CAvgInfoReader** používá instance externích tříd AVG ve své implementaci, proto je mezi ní a třídami závislost „uses“. [4] Třída **CAvgCfgSettings** přistupuje přímo ke konfiguračním souborům a její metody slouží k nastavování komponent antiviru na nižší úrovni. Má možnost komponentu zcela vypnout, nebo spustit a podrobněji nastavovat její nastavení. Třída **ComponentStateWatcher** slouží k nastavení stavu komponenty přes komponentu WatchDog antiviru AVG.[14] Dokáže komponentu pouze aktivovat, pokud je deaktivovaná. Třída **ComponentStateSetting** má metody pro spuštění skenu počítače a provedení update nebo upgrade antiviru v případě že je stará virová databáze, nebo stará verze AVG.

O interakci s uživatelem se stará agent NAP sám, protože agent SHA běží v systému jako služba a nemá tedy možnost komunikovat s uživatelem. NAP agent pouze zobrazuje informace o stavu klienta ve spodní liště okna. Zobrazuje tři stavy informující o potřebě opravy klienta a o průběhu automatické opravy.[7]



3.1.1 Diagram tříd agent SHA a knihovny AvgInfoLib. [9]

[9] Zdroj: vlastní.

3.2 Implementace agenta SHA

Kapitoly o implementaci agenta SHA vychází především ze zdroje[10], kde je uveden příklad implementace.

3.2.1 Instalace a spuštění služby SHA agenta

Samotný SHA agent běží jako služba systému Windows.[7] Najdeme ji tedy po instalaci v service manažeru operačního systému. (services.msc)

Ovládání pomocí parametrů:

parametr `"install"`

Parametr install je volán pouze jednou při instalaci služby SHA agenta. Služba se po instalaci parametrem install spouští automaticky s každým startem systému Windows. Pro správný běh služby je potřeba zajistit rovněž automatický start služby Network Access Protection agent.

Implementace instalace aplikace jako služby systému Windows:

1. Kontrola zda služba již není nainstalovaná

   ```
   if (AvgShaService::IsInstalled())
   ```

2. Pokud služba není v systému – instalace služby

   ```
   AvgShaService::InstallSvc()
   ```

Otevření SCM – Service Manager[9]

```
serviceManager.Open(0, AvgWinServiceHandleBase::SAR_CREATE_SERVICE);
```

Vytvoření služby[9] :

```
service.Create(
                AvgShaServiceInternalName,
                AvgShaServiceName,
                AvgWinService::ServiceAccessRight(SC_MANAGER_ALL_ACCESS),
                AvgWinService::ST_WIN32_OWN_PROCESS |
                AvgWinService::ST_INETRACTIVE_PROCESS,
                AvgWinService:: SST_AUTO_START,
                AvgWinService::SEC_NORMAL,
                path.c_str(),
                AVG_TEXT(""),
                NULL,
                NULL,
                NULL,
                NULL
                );
```

Konstatnty uvedené jako řetězce identifikující službu v systému :

```
AvgShaServiceName = AVG_TEXT("Avg SHA service");
AvgShaServiceInternalName = AVG_TEXT("AvgShaService");
```

Parametr `"remove"`

Odstranění služby NAP SHA agenta ze systému. Tento parametr je volán pouze při odinstalaci služby ze systému.

Volání služby s tímto parametrem by mělo předcházet zastavení služby pomocí parametru `"stop"`. Po zavolání parametru remove bude služba odstraněna ze Service Manageru a nebude nadále automaticky spouštěna při startu počítače.

Implementace odstranění služby ze systému Windows:

1. Kontrola zda je služba nainstalována

   ```
   if (!AvgShaService::IsInstalled())
   ```

2. Volání funkce pro odstranění služby

   ```
   AvgShaService::RemoveSvc()
   ```

Otevření SCM – Service manageru pro odstranění služby [9]:

```
serviceManager.Open(0, AvgWinServiceHandleBase::SAR_DELETE);
```

Odstranění služby [9]:

```
AvgWinService service(serviceManager);
      if (service.Open(AvgShaServiceInternalName,
      AvgWinServiceHandleBase::SAR_DELETE)) {
                  service.Delete();
                  result = true;
      }
```

Parametr `"start"`

Spuštění nainstalované služby AVG SHA agenta.

Aplikace je volána s tímto parametrem jen v případě, kdy není spuštěná automaticky se startem systému. Standartně by se služba měla spouštět automaticky. Pouze v případě instalace je tento parametr použitý pro spuštění nainstalované služby aby se spustila okamžitě a nebylo třeba čekat až do restartu operačního systému. Při dalším spuštění operačního systému bude již služba spuštěna automaticky.

Parametr `"stop"`

Zastavení spuštěné služby AVG SHA agenta.

Aplikace je volána s tímto parametrem, například během odinstalace aplikace AVG SHA agenta. Před použitím parametru remove je vhodné použít parametr stop pro zastavení služby a dealokaci jejich prostředků a správné uvolnění COM rozhraní.

Parametr `"console"`

Parametr pro ladění aplikace a zobrazení chybových stavů.

Pomocí tohoto parametru se aplikace AVG SHA agenta spouští jako běžná konzolová aplikace. Použití je výhradně pro ladění aplikace, popřípadně kontrolu funkčnosti NAP. V tomto režimu agent vypisuje spoustu informací o běhu aplikace. Například napojení na NAP agenta, odesílání informací, příjem informací od serveru, seznam chybných komponent.

Instalace této služby bude probíhat automaticky a bude součástí instalačního balíčku antiviru AVG.

Implementace v souborech: Sha.cpp, Sha.h (viz. Příloha 2)

3.2.2 Registrace SHA agenta v systému

Registrací je myšleno napojení SHA agenta na službu NAP běžící v rámci systému. Připojení probíhá pomocí rozhraní COM a ATL[3].
SHA agent již musí být zaregistrován jako služba v systému. A služba NAP musí být spuštěná, abychom mohli službu AVG SHA agent k této službě zaregistrovat. Jedná se v podstatě o předání ID službě NAP aby „věděla" o našem SHA agentovi, který bude předávat informace o stavu počítače službě NAP, která je zašle k validaci na server[7].
Základní kroky[10] :

- Inicializace COM rozhraní pomocí funkce `CoInitializeEx`
- Nastavení COM pro přístup ke službě NAP, která běží jako NetworkService. Pomocí objektu `CsecurityDescriptor` a jeho metody `Allow`.
- Vytvoření instance rozhraní `InapClientManagement`
- Vyplnění registračních informací ve struktuře `NapComponentRegistrationInfo`
- Vyvolání metody `InapClientManagement->RegisterSystemHealthAgent` které registrační informace předáme.

  Komponenta je v tento moment zaregistrována jako SHA agent v operačním systému.

3.2.3 Připojení SHA agenta ke službě agent NAP

Informace v této kapitole vychází především ze zdroje[10].
Zaregistrování callback funkcí implementovaných v SHA, které služba NAP volá.

- Vytvoření instance rozhraní `CLSID_NapSystemHealthAgentBinding` funkcí `CoCreateInstance`
- Inicializace Callback funkcí pomocí `Initialize` a předáním ID SHA agenta

Inicializace a připojení SHA agenta k systému jsou realizovány v souborech:

Sha.cpp, Sha.h (viz. Příloha 2)

ShaModule.cpp, ShaModule.h (viz. Příloha 2)

3.2.4 Komunikace SHA agenta se službou agent NAP

Informace v této kapitole vychází především ze zdroje[10].

Komunikace probíhá pomoci služeb RPC, kdy služba NAP volá sadu callback funkcí implementovaných v SHA agentovi. Implementace SHA agenta spočívá především v implementaci rozhraní **INapSystemHelathAgentCallback**. Po zaregistrování SHA agenta jsou tyto callback funkce volány službou NAP.[7]

```
STDMETHOD(GetSoHRequest)(
        IN INapSystemHealthAgentRequest* pRequest) throw ();


STDMETHOD(ProcessSoHResponse)(
        IN INapSystemHealthAgentRequest* pIRequest) throw ();


STDMETHOD(NotifySystemIsolationStateChange)(void) throw ();


STDMETHOD(GetFixupInfo)(
        OUT FixupInfo** ppstatus) throw ();

STDMETHOD(CompareSoHRequests)(
        /* in */ const SoHRequest* lhs,
        /* in */ const SoHRequest* rhs,
        /* out */ BOOL* isEqual
        ) throw ();

STDMETHOD(NotifyOrphanedSoHRequest)(
        /* in */ const CorrelationId* correlationId
        ) throw ();
```

Odesílání informací

```
HRESULT FillSoHRequest(INapSoHConstructor** ppISohRequest) throw();
```

V této funkci se vytváří zpráva SoH, která bude odeslána na server k validaci.Tato SoH obsahuje informace o stavu komponent klienta a čas od posledního update.[7]

Je třeba vyplnit ID SHA Agenta, podle tohoto ID bude zpráva předána příslušnému SHV na serveru. ID je tudiž shodné s ID validačního AVG SHV (viz.dále)

```
value.vendorSpecificVal.vendorId = AvgNapSystemID;
```

Dále je třeba vyplnit SoH daty, které chceme odeslat.Velikost dat je teoreticky limitována. Limit se může lišit v závislosti na verzi operačního systému, nebo protokolu NAP samotného.[7] V každém případě je pro datové struktury více než dostačující (aktuálně kolem 2MB).

Funkce zajišťující aktuální informace o stavu antiviru AVG , vyplní těmito informacemi strukturu `AVG_STATE_DATA`, která je následně převedena do dynamické proměnné typu vektor. Každá informace je uložena zvlášť jako záznam vektoru.

Tento vektor uchovává položky ve formátu struktury `SAvg_All_State`.

```
struct SAvg_All_State{

            AvgInt32 compID;
            CAvgInfoReader::SAvgComponentStateSimple component;

      };
```

Kde `SAvgComponentStateSimple` je struktura

```
struct SAvgComponentStateSimple
      {

            AvgInt32 state;

            AvgTime timeInfo;

            AvgDWord dwVersion;

      };
```

A vektor `AvgSendBuffer` tedy vypadá následovně:

```
vector <struct CAvgInfoReader::SAvg_All_State> AvgSendBuffer;
```

Každá komponenta má svou strukturu `SAvg_All_State` a své ID uloženo ve této sktruktuře a svůj stav uložený v podstruktuře `SAvgComponentStateSimple`.

Vektor naplněný temito strukturami je postupně procházen a každá struktura se přidá k SoH datům. Odesílané data tedy vypadají jako SoH packet, který ve své datové části obsahuje určité množství zpráv uložených zasebou.[7] Počet závisí na počtu informací zjišťovaných SHA Agentem z antiviru.

Důvodem použití zvlášť struktury pro každou komponentu nebo informaci je variabilita a rozšířitelnost systému. Řeší problémy kompatibility novějšího klienta a staršího serveru a naopak. Server odpoví a validuje pouze zprávy, které zná.[7] Pokud přibude na SHA klientovi o informaci navíc, bude ji starší server ignorovat a odpovídat na pouze jemu známé informace.
Vytvořený vektor naplněný informacemi, se postupně prochází a pomocí funkce `AppendAttribute` Se přidává každá informace na konec SoH packetu.

```
for ( AvgSendBuffer_it=AvgSendBuffer.begin() ; AvgSendBuffer_it <
AvgSendBuffer.end(); AvgSendBuffer_it++ ){

//ID (vysvětleno výše)
value.vendorSpecificVal.vendorId = AvgNapSystemID;
//Velikost odesílané zprávy
value.vendorSpecificVal.size = sizeof(struct SAvg_All_State);
//Samotné data zprávy
value.vendorSpecificVal.vendorSpecificData = (PBYTE)&(*AvgSendBuffer_it);

//Přidání informace na konec SoH packetu
(*ppISohRequest)->AppendAttribute(sohAttributeTypeVendorSpecific,&value);}
```

Příjem informací

Další důležitou Callback funkcí je funkce pro příjem odpovědi od serveru.

```
HRESULT HandleSoHResponse(INapSoHProcessor * pSohProcessor,
                                    SystemHealthEntityId  systemHealthId,
                                    BOOL doFixup) throw();
```

Tato funkce je volána, pokud je potřeba příjmout a zpracovat odpověd od serveru. Funkce přímá všechny informace, které jsou serverem generovány.[10] Jsou to struktury

```
//Structure of incomming data from validation server
struct AVG_RESPONSE_DATA {
      AvgInt32 ComponentID;
      AvgInt32 State;
};
```

Obsahující informace o komponentách které jsou ve špatném stavu, popřípadně příkaz k update. Příjem dat probíhá následovně. Je zavolána Callback funkce `HandleSoHResponse`, která jako parametr příjme ukazatel na přijatou zprávu a pomocí funkcí rozhraní `INapSoHProcessor` přjaté data rozparserujepomocí funkce `FindNextAttribute` na struktury typu `AVG_RESPONSE_DATA`.

```
struct AVG_RESPONSE_DATA {
      AvgInt32 ComponentID;
      AvgInt32 State;
      char info[30];
};
```

Přijaté data můžou obsahovat libovolný počet těchto zpráv, počet zavisí na počtu informací odeslaných serverem. Počet zpráv odeslaných serverem se dá zjistit zavoláním `GetNumberOfAttributes(&AttributeCount)`. Obecně tato struktura obsahuje ID komponenty, která je ve špatném stavu a její stav. Popřípadně může obsahovat příkaz k update a v položce info volitelné informace jako například IP adresu.

Tyto a další Callback funkce jsou implemntovány v souborech

`Callback.cpp, Callback.h` (viz. Příloha 2)

3.3 Komunikace s antivirem AVG

Návrh klientské strany je rozdělen na dvě části. V předchozích kapitolách byl uveden návrh a implementace agenta SHA, který se stará o komunikaci. Tato kapitola se bude zabývat knihovnou AvgInfoLib, která má na starost komunikaci s produktem AVG.
Většina informací uvedených v podkapitolách této kapitoly vychází ze softwarové dokumentace společnosti AVG. Reference zdrojů uvedené v textu jsou dosažitelné pouze v rámci interních informací AVG a nejsou veřejně dostupné. Kapitola vychází především ze zdrojů [13],[14] a [15].

3.3.1 Sběr dat z antiviru AVG

Vytvoření třídy pro přístup ke komponentě WatchDog antiviru AVG[14] a stažení potřebných informací.

- Připojit se ke komponentě WD
- Přečíst informace o všech komponentách antivirového programu

Informace o stavu klienta jsou čteny z komponenty WatchDog. Komponenta monitoruje stav ostatních komponent antiviru a poskytuje nejaktuálnější informace o jeho stavu. Připojením k této komponentě lze získat informace o aktuálním stavu všech komponent, stáří virové databáze, poslední čas update, verzi antiviru a čas posledního provedeného skenování počítače.[14]
Pro příjem informací o konkrétní komponentě je implementována funkce `CAvgInfoReader::ReadComponentStateFromWD`, která vrací informace ve struktuře

```
struct SAvgComponentState
      {     //! \brief State int32 value
            AvgInt32 state;
            //! \brief State text describing the state.
            AvgString stateText;
            //! \brief Version of the component.
            AvgString versionString;
            //! \brief Time info.
            AvgTime timeInfo;
            //! \brief DWord value
            AvgDWord dwVersion;
            //! \brief Constructor.
      };
```

Data jsou čteny z komponenty WatchDog pouze na žádost serveru, což nastává ve dvou situacích:

1. Připojení k serveru, server žádá informace o stavu klienta - Prvotní připojení klienta k serveru
2. Změna stavu libovolné komponenty klienta - Některý z hlídačů informoval server o změně stavu klienta a server si tedy vyžádal nové informace

Implementováno v AvgInfoLib\AvgInfo.cpp, AvgInfoLib\AvgInfo.h (viz. Příloha 2)

3.3.2 Kontrola změny stavu komponent antiviru AVG

Vytvoření hlídače změny stavu komponenty a zaregistrování pro všechny komponenty antiviru. Při změně stavu komponenty je třeba vyvolat funkci `NotifySoHChange()`, která upozorní službu NAP na změnu stavu komponent a vyšle nový požadavek k serveru na validaci klienta s novým stavem.[1]

Pro zaregistrování hlídače pro všechny komponenty antiviru je třeba použít metodu `RegisterWatcherForAll()` třídy `ComponentStateWatcher`.

Při inicializaci této třídy se vytvoří seznam komponent pro které bude hlídač zaregistrován.[14]

Jednoduchým přidáním komponenty do konstruktoru třídy tedy přidáme watcher na libovolnou komponentu :

```
CWDStateWatcher *pwdStateWatcher ;
      pwdStateWatcher   = new CWDStateWatcher(SCHEDULER_KEY_NAME);
```

Kdy funkce `RegisterWatcherForAll()`

Prochází vektro vytvořený v inicializaci a registruje Watcher pro každou komponentu uloženou ve vektoru.

```
void ComponentStateWatcher::RegisterWatcherForAll(){

for ( m_watcher_it=m_watcher.Begin() ; m_watcher_it != m_watcher.End();
m_watcher_it++ ){
            m_errorCode=(*m_watcher_it)->Register();
      }
}
```

Při změně stavu libovolné komponenty se volá callback funkce [14]

`void ComponentStateWatcher::CWDStateWatcher::OnValueChanged`

která volá funkci SHA agenta `OnStateChanged()` která znovu načte aktuální stav všech komponent a zavolá funkci `NotifySoHChange()` rozhraní `INapSystemHealthAgentBinding`. Zavoláním této funkce informuje server o změně stavu některé komponenty a ten si pak vyžádá informace o stavu všech komponent a provede znovu validaci stavu klienta. Proběhne odeslání nových informací z klienta na server.

```
void CAvgStateCallback::OnStateChanged(){

      AVG.SetNewState(); //Nacteni novych stavu
      binding->NotifySoHChange();//informovani serveru o zmene stavu

}
```

Implementováno v AvgInfoLib\ComponentStateWatcher.cpp, *.h (viz. Příloha 2)

3.3.3 Přijetí informací o chybách

Tato kapitola vychází především z interních zdrojů [13] a [14].

Když server zpracuje přijaté informace od klienta, rozhodne se, zda ho připustí k síti nebo ne. Pokud klient nebude připuštěn k síti, znamená to, že některá z jeho komponent není v požadovaném stavu. Například vypnutý rezidentní štít AVG. Server vrátí zpět klientovi informace o komponentách, jejichž stav není v souladu s politikou serveru.[7]

Struktura přijatých dat je zvlášť struktura pro každou chybu:

```
struct AVG_RESPONSE_DATA {
      AvgInt32 ComponentID;
      AvgInt32 State;
      char info[30];
};
```

Pro každou komponentě, která nesplní požadavky serveru, bude poslána tato struktura (zpráva) informující klienta o chybě na dané komponentě.

`AvgInt32 ComponentID` – ID komponenty, pro kterou byla chybná zpráva vygenerována.

Konstanty identifikující jednotlivé komponenty jsou brány ze seouboru konstant \..krnl\intf\wddefs.h .Například `ANTIVIRUS_KEY_ID.[13]`

Server nezasílá klientovi pouze informace o chybách komponent, ale rovněž příkaz k Update, Upgrade klienta v případě kdy má klient zastaralou databázi nebo verzi antiviru, popřípadně dobu timeout, po které je klient nucen znovu odeslat svůj stav a znovu obnovit připojení. Tyto konstanty použitelné jako ID zpráv jsou uloženy v souboru Common.h na straně klienta a v souboru ShvCommon.h na strane serveru.

```
UPDATE_WITH_PARAMETER
UPDATE_NO_PARAMETER
UPGRADE_WITH_PARAMETER
UPGRADE_NO_PARAMETER
TIME_LAST_UPDATE
TIMEOUT
```

Při příjmu zpráv `UPDATE_WITH_PARAMETER a UPGRADE_WITH_PARAMETER` je v proměnné info uložena IP adresa serveru, odkud se bude update provádět.

Při příjmu zprávy od serveru se podle ID vyhodnotí o jakou komponentu nebo příkaz se jedná. V případě přijetí chyby o komponentě se klient snaží tuto komponentu spustit, pokud neběží a nastavit ji do stavu akceptovaného politikou serveru. A to zaslání zprávy ke komponentě WatchDog a zápisem do konfiguračního souboru.

Spuštění komponenty

Tato kapitola vychází především z interních zdrojů [13] a [14].

Pro spuštění komponenty, která v antiviru ani neběží, nebo je pozastavena se provádí pomocí komponenty WatchDog. Zasláním zprávy s konkrétním příkazem směrem ke komponentě WatchDog jsme schopni spustit libovolnou komponentu.[14]

Po přijetí chyby o konkrétní komponentě se nejprve snažíme danou komponentu spustit:

```
ComponentStateSetting::SendCommand(WD_COMMAND_START_RS);
```

Příkaz `WD_COMMAND_START_RS` odeslaný k WD spustí komponentu Resident Shield .

Zasíláni zpráv je implementováno ve třídě `ComponentStateSetting`. V inicializaci této třídy dochází k napojení ke komponentě WatchDog pomocí vytvoření kanálu nad asociativní pamětí komponenty.

```
// initialize associative memory
AVG_TEST_SUCCEEDED_ASSERT(m_assMemory.Initialize(Avg8CommRootConnectionPar
ameters::GetConnectionParameters()));

// create communication channel
AVG_TEST_SUCCEEDED_ASSERT(m_pChanell.Attach(new
AvgBasCommAssociativeMemoryChannel(m_assMemory)));

// assemble path to server plugin component command channel
AvgString commandChannelPath;
commandChannelPath.Format (AVG_TEXT("/%s/%s/%s"), COMPONENTS_KEY_NAME,
WD_KEY_NAME, SYSTEM_COMMAND_VALUE_NAME);

// create command channel instance
AVG_TEST_SUCCEEDED_ASSERT(m_pChanell-
>Open(commandChannelPath.GetValue()));
```

Po této inicializaci je možné zasílat příkazy komponentě WatchDog, která tyto příkazy vykonává nad komponentami antiviru. Ovšem použití není pouze ke spuštění komponent, ale používá se i ke spuštění update nebo upgrade.

Implementováno v AvgInfoLib\ComponentStateSetting.cpp,*.h (viz. Příloha 2)

Nastavení komponenty

Tato kapitola vychází především z interních zdrojů [13]a [15].

Přímo samotné nastavení stavu komponenty se provádí zápisem do konfiguračního souboru antiviru.

Zápis do konfiguračního souboru je implementován ve třídě `CAvgCfgSetting`.

`CAvgCfgSetting::Initialize()`

Ještě před samotným zápisem do konfiguračního souboru antiviru je třeba si stáhnout konkrétní soubor konfigurací pro danou komponentu.Tyto informace jsou uvedeny v souboru \..\tables\cfg_genconf.h

Pro většinu komponent si stačí stáhnout konfiguraci kernelu antiviru :

```
AVG_TEST_THROW(m_pConfigManager->getConfig(CKEYROOTID_krnl,__DEFAULT_FS__,
pSetupConfig.OutRef() ));
```

A nastavit práva přístupu :

```
AVG_TEST_THROW(pSetupConfig->setClientAccessRights(AVGCFG_RIGHT_ADMIN));
```

Ovšem pro nastavení komponent IDP a Firewall je třeba stáhnou konfiguraci zvlášť :

```
AVG_TEST_THROW(m_pConfigManager->getConfig(CKEYROOTID_idp,
__DEFAULT_FS__, pIdpConfig.OutRef() ));
AVG_TEST_THROW(m_pConfigManager->getConfig(CKEYROOTID_fw,  __DEFAULT_FS__,
pFwConfig.OutRef() ));
```

Zápis do konfiguračního souboru pro nastavení komponenty :

Příklad jednoduchého nastavení - RESIDENT-SHIELD
Pouze spuštění RS

```
m_config.pSetupConfig->putBoolValue(CKEYID_krnl_resident_enabled,true);
m_config.pSetupConfig->ApplyConfig();
```

Příklad složitějšího nastavení- LINK-SCANNER
Spuštění LS a jeho nastavení

```
m_config.pSetupConfig->putBoolValue(CKEYID_krnl_linkscanner_enabled,true);
m_config.pSetupConfig->putBoolValue
(CKEYID_krnl_networkScanner_httpScanner_safesurf,true);
m_config.pSetupConfig->putBoolValue
(CKEYID_krnl_linkscanner_reporting,true);

m_config.pSetupConfig->ApplyConfig();
```

Implementováno v AvgInfoLib\ComponentCfgSetting.cpp,*.h (viz. Příloha 2)

Spuštění upgrade a update antiviru

Tato kapitola vychází především z interních zdrojů [13] ,[14] a [15].

Spuštění update a upgrade bez parametru se provádí zasláním zprávy ke komponentě WatchDog. Update, upgrade s parametrem se provádí samotným spuštěním aplikace update.exe s parametry. Po přijetí zprávy s ID `UPDATE_NO_PARAMETER` je odeslán příkaz, který spustí update antiviru AVG bez parametru. Tento příkaz provede interně spuštění update.exe souboru bez parametrů. Update probíhá ze standartním URL adres definovaných v nastavení antiviru AVG.

```
SendCommand(WD_COMMAND_START_UPDATE);
```

Po přijetí zprávy s ID `UPDATE_WITH_PARAMETER` je volána metoda `RunUpdateWithParam`, která sestaví příkaz ze zvolených parametrů a tímto příkazem spustí update.exe proces.

```
if(AVG_DATA->ComponentID==UPDATE_WITH_PARAMETER){
//Send command to start update with parameters

m_componentsSetting.RunUpdateWithParam(AVGAPI_UPDATE_SOURCE_INET,
(LPCWSTR)AVG_DATA->info,AVGAPI_UPDATE_PRIORITY_TYPE_PROGRAM,true);
}
```

Kde parametr `AVG_DATA->info` obsahuje IP adresu,ze které se bude provádět UPDATE,UPGRADE antiviru.Tato IP adresa je nastavena administrátorem na validačním serveru.Může se jednat o IP adresu některého z interních serverů sítě,na který bude mít klient přistup I v případě nepřijetí serverem a kde si může stáhnout potřebný update antiviru aby byl později serverem akceptován a připuštěn do sítě.

Implementováno v AvgInfoLib\ComponentStateSetting.cpp,*.h (viz. Příloha 2)

3.3.4 Přehled součástí

Komponenta SHA agenta je součástí antiviru AVG a je instalována a spouštěna přímo instalačním balíčkem antiviru. Pro testovací účely je návod ruční instalace a spuštění připraven v přílohách (viz. Příloha 1). Rovněž dokument pojednávající o postupu rozšíření SHA agenta v případě, že bude třeba odesílat více informací nebo se změní některé komponenty antiviru. (viz. Příloha 3)

Klientská část návrhu je tvořena třemi základními projekty. Tím prvním je knihovna **AvgInfoLib**, která implementuje třídy pro přístup k hlídači, konfiguračním souborům a hlídačům. Dalším projektem je projekt **Common**, který obsahuje pouze podpůrné funkce a především potřebné konstanty (ID SHA, ID komponent).
A posledním projektem je **Sha** – samotný SHA agent.

SHA agent
AvgShaService.cpp – implementace funkcí pro zavedení služby do operačního systému
Callback.cpp – implementace callback funkcí SHA agenta
Getavginfo.cpp – implementace třídy, která je přístupovým bodem ke knihovně AvgInfoLib. Třída obsahuje objekty všech tříd z projektu AvgInfoLib a přistupuje k jejím metodám.
Sha.cpp – zavádí aplikaci jako službu pomocí tříd z AvgShaService.cpp a registruje SHA agenta do systému pomocí funkcí z ShaModule.cpp.
ShaModule.cpp – funkce pro registraci SHA agenta v systému.

AvgInfoLib
AvgInfo.cpp – třída pro připojení ke komponentě WatchDog (hlídač) a načtení stavů všech komponent včetně zjištění verze programu a databáze.
ComponentCfgSetting.cpp – třída pro práci s konfiguračními soubory antiviru AVG.
ComponentStateSetting.cpp –třída pro zasílání zpráv ke komponentě WatchDog a spouštění UPDATE/UPGRADE.
ComponentStateWatcher.cpp – třída pro registraci hlídačů pro každou komponentu.

Common
Common.cpp – podpůrné funkce pro Callback.cpp
Common.h – konstanty (shodné s ShvCommon.h) pro vzájemnou komunikaci SHA agenta a serveru.

(viz. Příloha 2)

4 Analýza, návrh a implementace komponent serveru architektury NAP

Cílem je navrhnout komponentu SHV serveru NAP, která má za úkol podle přijatých informací o stavu klienta, vyhodnotit jeho stav jako odpovídající nebo neodpovídající politice sítě. Validátor SHV je vždy tvořen třetí stranou a může jich tedy být v systému více k různým produktům podporujícím NAP. SHV validátor je komponenta zaregistrovaná v administračním serveru, který ji předává data od konkrétního klienta. Tyto základní požadavky vychází jednak z architektury a možností použití technologie NAP[5] a rovněž z požadavků společnosti AVG.

4.1 Návrh validátoru SHV

Základní požadavky na validátor SHV vychází z jeho obecně dané architektury dokumentované společností Microsoft a způsobu nasazení v operačním systému na straně serveru. Seznam požadavků je dále rozšířen o požadavky společnosti AVG navazující na funkcionalitu serveru.
Základní požadavky na SHV validátor jsou[7]:

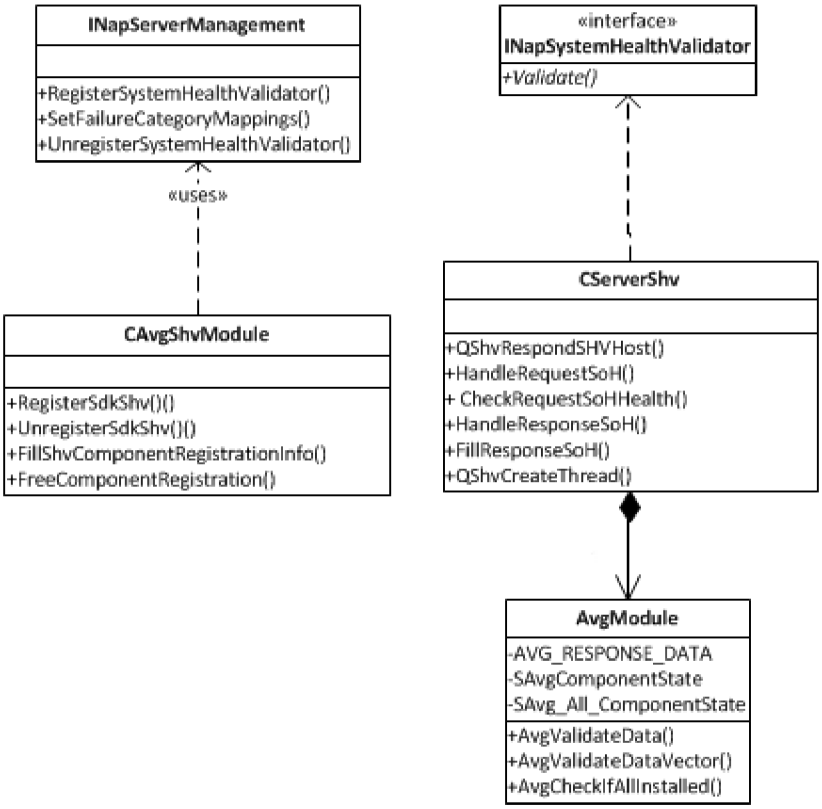
- Uložen knihovna DLL.
- Identifikátor SHV se musí shodovat s identifikátorem SHA komponenty pro antivir AVG.
- Musí být zaregistrován v administračním serveru.
- Komunikace se službou NPS.

Rozšířené požadavky podle požadavků společnosti AVG:

- Porovnávat přijaté informace s nastavenou politikou sítě pro konkrétní produkt.
- Musí obsahovat GUI pro nastavení požadovaných politik.
- Ukládat nastavení do XML souboru.
- Číst nastavení z XML souboru.
- Vždy mít k dispozici informace o nejnovějších verzích AVG.
- Možnost úprav validátoru v případě rozšíření počtu komponent antiviru.

Implementace SHV je rozdělena na část validátoru a na část grafického rozhraní pro nastavení politik sítě a uložení do XML souboru.

Pro zaregistrování validátoru na serveru existuje třída **CAvgShvModule**, která využívá instance třídy **INapServerManagement** aby mohla registraci provést. Dále pak musí každý SHV validátor implementovat rozhraní INapSystemHealthValidator obsahující jedinou metodu k implementaci a to metodu validate. Třída **AvgModule** validuje přijaté data vůči politice sítě a aktuální verzi antivirového software.



4.1.1 Diagram tříd validátoru SHV.[10]

[10] Zdroj: vlastní.

4.2 Implementace validátoru SHV

Kapitoly o implementaci validátoru SHV vychází především ze zdroje[11] a [12], kde je uveden i příklad implementace z dokumentace MSDN.

4.2.1 Spuštění a zastavení SHV validátoru v systému

Spuštění a registrace komponenty

SHV validátor je komponenta kompilovaná do knihovny DLL, kterou je třeba do systému zaregistrovat pomocí spuštění programu regsvr32 s parametrem:

Instalace - regsvr32 Shv.dll

Tento postup instalace není třeba používat, je obsažen v instalačním balíku, který se sám stará o spuštění DLL knihovny. Použití pouze v případě ruční instalace SHV validátoru.

Dalším krokem je registrace SHV validátoru v operačním systému serveru.

Registrací se objeví SHV validátor v komponentě NPS .

Funkce `CAvgShvModule::RegisterSdkShv`

Vytvoří instanci rozhraní `InapServerManagement` funkcí `CoCreateInstance.`

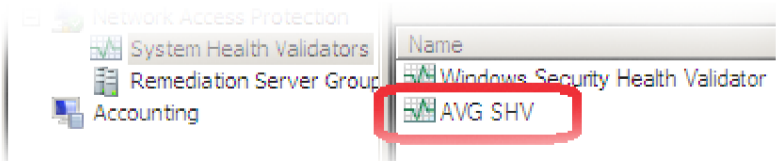
```
CoCreateInstance(CLSID_NapServerManagement,NULL,CLSCTX_INPROC_SERVER
)
```

- Vyplní strukturu `NapComponentRegistrationInfo` informacemi o vydavateli (AVG) a ID SHV serveru. ID se musí shodovat s ID SHA komponenty pro antivir AVG
  `FillShvComponentRegistrationInfo(&shvInfo);`

- Zavoláním funkce `RegisterSystemHealthValidator` pro registraci SHV validátoru. Jako parametr je předán ukazatel na třídu virtuálních funkcí implementovaných callback funkcí využívaných službou NAP pro vložení dat od klienta k validaci a ke čtení výsledků validace.

```
RegisterSystemHealthValidator(&shvInfo,(CLSID*)&__uuidof(CServerShv)
```

V případě úspěšného zaregistrování komponenty v systému, se SHA validátor ihned zobrazí na dialogu komponenty NPS.[6]



4.2.1.1AVG SHV zaregistrován v systému[11]

[11] Zdroj: vlastní.

Zastavení a odregistrování komponenty

SHV validátor lze jednoduše ze systému odstranit pomocí UnloadDLL funkce aplikace regsvr32 :

Odinstalace - regsvr32 /u Shv.dll

Tento postup instalace není třeba používat, je obsažen v instalačním balíku, který se sám stará o odstranění DLL knihovy. Použití pouze v případě ruční odinstalace SHV validátoru.

Dalším krokem je odregistrace SHV validátoru v operačním systému serveru.

Odregistrováním se smaže SHV validátor z komponenty NPS .

Funkce `CAvgShvModule::UnregisterSdkShv()`

- Vytvoří instanci rozhraní `InapServerManagement` funkcí `CoCreateInstance.`
- Zavoláním funkce `UnregisterSystemHealthValidator` pro odregistraci SHV validátoru. Jako parametr je předáno ID `AvgNapSystemID` definované v souboru `ShvCommon.h`

  ```
  static const UINT32 AvgNapSystemID = 0x000137F0;
  ```

  volání funkce

  ```
  UnregisterSystemHealthValidator(AvgNapSystemID);
  ```

V případě úspěšného odregistrování komponenty v systému se SHA validátor ihned smaže z dialogu komponenty Network Policy Server.

Registrace a odregistrace je

implementována v souboru s \NAP\SHV\SHV\ShvModule.cpp,*.h (viz. Příloha 2)

4.2.2 Přijetí dat od klienta k validaci

Pro každého klienta žádajícího o validaci se vytvoří samostatné vlákno, které rozhodne o jeho stavu. Funkce implementace `QshvCreateThread` má za úkol vytvořit vlákno. Sdílení dat mezi vláknem a rodičem funguje pomocí funkce `AsyncThreadHandler`, která vláknu zpřístupní datové sdílené struktury.[12] Implementováno v ShvModule.cpp,*.h (viz. Příloha 2)

Pro každý přijatý SoH paket od klienta je volána Callback funkce HandleRequestSoH, které je celý SoH packet předán.

Funkce HandleRequestSoH příjme data od klienta a předá je funkci `CheckRequestSoHHealth` pro validaci.[11]

Úkolem funkce `CheckRequestSoHHealth` je získané data rozparserovat na struktury `struct SAvg_All_State` , které jsou ukládány do vektoru, který je následně předán funkci `AvgValidateDataVector` pro validaci informací o klientovi.

`CServerShv::CheckRequestSoHHealth() :`

- Nejprve zjistíme počet přijatých zpráv obsažených v SoH.Počet se liší a závisí na počtu infomací které odesílá klient.Pro každou komponentu je jedna zpráva.

```
hr = pSohRequest->GetNumberOfAttributes(&AttributeCount);
```

- Postupné procházení přijatého SoH paketu a vybírání zpráv

```
pSohRequest->FindNextAttribute(i, sohAttributeTypeVendorSpecific,
&index);
```

- Uložení přijaté zprávy do vecotru pro validaci

```
RecvBuffer.push_back((const AvgData::SAvg_All_State &)(*pAttrValue-
>vendorSpecificVal.vendorSpecificData));
```

Přijaté data obsahují zprávy (struktury) pro každou komponentu :

```
struct SAvg_All_State
{
      AvgInt32 compID;
      SAvgComponentStateSimple component;

};

struct SAvgComponentStateSimple
{
         AvgInt32 state;
         AvgTime timeInfo;
         AvgDWord dwVersion;
};
```

Všechny tyto přijaté zprávy máme uložené ve vectoru RecvBuffer.

Implementováno v ShvCallback.cpp,*.h (viz. Příloha 2)

4.2.3 Načtení politiky sítě

Načtení politik serveru je krok před samotnou validací klienta. Klient se validuje vůči politice serveru. V politice serveru je uložen seznam komponent a stav v jakém se komponenta musí nacházet, aby byl klient do systému připuštěn.
Celá konfigurace je uložena v podobě XML souboru. XML soubor je tedy nutné rozparserovat a uložit data do struktury, která bude následně použita pro validaci klienta.

Nahraní konfigurace z XML

Pomocí rozhraní společnosti Microsoft DOM načteme konfigurační XML soubor a rozparserujeme podle značek. Hodnoty se ukládají do struktury.
Struktura konfiguračního souboru politik AvgShvConfig.xml
Po parserování XML souboru se politika uloží do struktury

```
struct ShvXmlConfig {
        ShvXmlComponent AntivirusConf;
        ShvXmlComponent RSConf;
        ShvXmlComponent FWConf;
        ShvXmlComponent AMConf;
        ShvXmlComponent ARConf;
        ShvXmlComponent AdminClientConf;
        ShvXmlComponent AnalysisConf;
        ShvXmlComponent AntispamConf;
        ShvXmlComponent AntispyConf;
        ShvXmlComponent EMSConf;
        ShvXmlComponent LicenseConf;
        ShvXmlComponent LinkScannerConf;
        ShvXmlComponent NetworkScannerConf;
        ShvXmlComponent IDPConf;
        ShvXmlComponent UpdateMgrConf;
        // server plugins
        ShvXmlComponent SrvExchAntiSpamConf;
        ShvXmlComponent SrvExchRoutingTAConf;
        ShvXmlComponent SrvExchSmtpTAConf;
        ShvXmlComponent SrvExchVSAPIConf;
        ShvXmlComponent SrvSharePointConf;
        //Updates
        ShvXmlUpdate UpdateConf;
        ShvXmlTime TimeOfLastUpdate;
        ShvXmlVersion AvgVersion;
        ShvXmlTime TimeOut;

};
```

Nastavení politiky pro jednotlivé komponenty

Komponenty mohou nabývat 4 stavů:

- **Running** – stav komponenty je „běžící“ aktualizovaná a plně funkční
- **Warning** – komponenta je v AVG nainstalována,ovšem je v chybovém stavu. Buď je deaktivována, nebo není plně aktualizována.
- **Error** – komponenta nejspíš není v AVG nainstalována, nebo je zcela zastavena a je v chybovém stavu.
- **Ignore** – stav komponenty je zcela ignorován (podobný stavu error)

Reprezentace konfiguračního souboru ve formátu **XML:**

```
<?xml version="1.0"?><!--AVG SHV validator configuration file.--><AvgShvConfig>
        <Anti-Virus>
                <state>ignore</state></Anti-Virus>
        <Anti-Spy>
                <state>warning</state></Anti-Spy>
        <Resident-Shield>
                <state>running</state></Resident-Shield>
        <Firewall>
                <state>running</state></Firewall>
        <Alert-Manager>
                <state>ignore</state></Alert-Manager>
        <Anti-Rootkit>
                <state>warning</state></Anti-Rootkit>
        <Network-Scanner>
                <state>ignore</state></Network-Scanner>
        <Link-Scanner>
                <state>warning</state></Link-Scanner>
        <Anti-Spam>
                <state>ignore</state></Anti-Spam>
        <Email-Scanner>
                <state>error</state></Email-Scanner>
        <Update-Manager>
                <state>running</state></Update-Manager>
        <ID-Protection>
                <state>ignore</state></ID-Protection>
        <License>
                <state>ignore</state></License>
        <Analysis>
                <state>warning</state></Analysis>
```

Příklad konfigurace komponenty:

```
<Resident-Shield>
   <state>running</state>
</Resident-Shield>
```

Klient žádající o připojení musí mít komponentu rezidentní štít plně funkční a aktualizovanou. V opačném případě bude vrácen chybový stav a klient označen za nevyhovující.

```
<Resident-Shield>
   <state>error</state>
</Resident-Shield>
```

Klient nemusí mít komponentu dokonce ani nainstalovanou v AVG a bude shledán jako vyhovující.

Nastavení politiky pro jiné typy zpráv:

```
<Update>
        <state>auto</state>
        <IP>www.google.com</IP></Update>
<Update-LastTime>
        <state>ignore</state>
        <time>24</time></Update-LastTime>
<Avg-Version>
        <state>ignore</state>
        <version>9.555</version></Avg-Version>
<Time-out>
        <state>check</state>
        <timeout>4</timeout></Time-out>
```

Konfigurace Update:

```
<Update-LastTime>
  <state>24</state>
</Update-LastTime>
```

Číslo znamená nejstarší možná virová databáze **v hodinách**, kterou klient může mít, aby byl ještě schválen jako vyhovující.

```
<Update>
  <state>manual</state>
  <IP>192.168.0.1</IP>
</Update>
```

Nastavení manual znamená, že antivir AVG bude instruován vykonat update ne ze serveru AVG, ale z některého serveru v síti specifikovanou parametrem IP, který nabízí update programu AVG. Při konfiguraci **auto** bude antivir AVG instruován k vykonání automatického update z adres nastavených v jeho konfiguraci.

Konfigurace nastavení Avg-Version. Klient musí mít nejnovější verzi antiviru AVG, pokud je tato politika nastavená na **check**. Nastavení má dvě volby **check** a **ignore** a číslo aktuální verze AVG. Číslo verze je nastaveno a uložena automaticky. Server stahuje informaci o nejaktuálnější verzi antiviru.

```
<Avg-Version>
   <state> check </state>
   <version> 2456879 </ version >
</Avg-Version>
```

Konfigurace Time-out je stejná jako Avg-version. Číslo uložené v uzlu timeout udává počet hodin do timeout.

```
<Time-out >
  <state> check </state>
  <timeout> 0 </ timeout >
</ Time-out >
```

Implementováno v XmlConfReader.cpp,*.h (viz. Příloha 2)

4.2.4 Načtení aktuálních verzí antiviru AVG

Funkce serveru, která zjišťuje aktuální vydanou verzi AVG. Administrátor má možnost tuto volbu nastavit tak, aby byla verze klienta vždy validována vůči nejaktuálnější verzi AVG. Klient bude instruován k automatickému upgrade svého antiviru.

Soubor aktuálních verzí: http://shadow.grisoft.cz/beta/90/update/avg9infowin.ctf

Nejdůležitější položka určující číslo nejnovější verze je položka `w9all826mo`.číslo verze je 826.

Celá verze má číslo 900826.

```
bin(w9all826mo.bin)grp(admcl:825;am:782;arkt:782;arkta:782;arktx:778;aspm
:826;aspma:826;aspmc:778;chjc:794;chjca:826;chjcx:826;core:826;corea:826;c
orex:778;emc:782;eml:794;eseex03:782;eseex07:820;esesp:820;eseui:782;esevs
api:825;fc1:816;fw:825;fwd:677;fwd6:677;ids:782;idsvta:778;idsvtx:778;idsw
7a:778;idsw7x:778;idsxpx:778;ima:783;imx:783;krnl:820;krnla:820;lng:808;ln
gcz:826;lngda:826;lngfr:826;lngge:826;lnghu:826;lngid:826;lngin:826;lngit:
826;lngjp:826;lngko:826;lngms:826;lngnl:826;lngpb:826;lngpl:826;lngpt:826;
lngru:826;lngsc:826;lngsk:826;lngsp:826;lngtr:826;lngus:826;lngzh:826;lngz
t:826;ls:820;lsa:820;lsff:825;lsie:825;lsiea:825;lsimg:778;ns:782;nsa:820;
nsx:820;ofc:782;rdsta:782;rsa:820;rsx:820;setup:826;smgr:793;st:782;sta:78
2;tb:825;tdi:778;tdia:820;tdix:820;ui:825;upd:805;vwsc:782)pla(w95+)var(fu
ll)tm(1005050759)len(65690297)
```

Získání souboru obsahující informace o verzích pomocí metody GET protokolu HTTP:

```
string text = "GET /beta/90/update/avg9infowin.ctf HTTP/1.1\r\n";
    text += "Host: shadow.grisoft.cz\r\n";
    text += "Connection: keep-alive\r\n";
    text += "Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n";
text += "\r\n";
```

Provede se připojení pomocí soketu k síti a stažení souboru avg9infowin.ctf.

Pokud není připojení k internetu dostupné, je použita poslední známá verze souboru, která je uložená na disku serveru. Klient je tedy vždy validován vůči nejnovější možné verzi i v případě kdy není dostupné připojení k internetu.

4.2.5 Validace stavu klienta

Po přijetí dat od klienta a načtení politiky serveru ze souboru XML jsme schopni porovnat klienta vůči nastaveným bezpečnostním politikám.

Srovnání klienta probíhá ve dvou fázích. První fáze porovnává přijaté informace od klienta s politikou nastavenou pro každou komponentu na serveru. Ve druhém kroku server testuje nepřijaté informace. Pokud neodešle informaci o určité komponentě, server bere komponentu jako nenainstalovanou a podle konfigurace zjišťuje, zda absence komponenty na klientské stanici má, či nemá vliv na bezpečnost. Pokud nemá klient nainstalovanou komponentu, která je politikou serveru vyžadována, nebude připuštěn do sítě.

Z předchozích kroků je konfigurace serveru načtena v e struktuře `struct ShvXmlConfig`.

První fáze:

Kontrola přijaté zprávy funkcí

`AvgValidateDataVector()` která příjme jako argument jednu přijatou zprávu, vyhodnotí porovnání jejího stavu s nastavenou politikou pro tuto komponentu,v kladném případě nehlásí nic,v případě neshody nahlásí chybu a uloží přijatou zprávu do vectoru `m_avgresponsvector`, který je následně po validaci všem komponent použit pro odeslání chybových informací.

Příklad pro Antivirus komponentu :

```
if(RecvBuffer->compID == ANTIVIRUS_KEY_ID){
       //Test shody stavu prijate komponenty s politikou
       if(RecvBuffer->component.state < config.AntivirusConf.state ){
            //V pripade neshody :
            UnhealthyState=1;

            AVG_RESPONSE_DATA *AntivirusState = new AVG_RESPONSE_DATA;
            AntivirusState->ComponentID=ANTIVIRUS_KEY_ID;
            AntivirusState->State=RecvBuffer->component.state;
            m_avgresponsvector.push_back(*AntivirusState);
            delete AntivirusState;
      }
      }
```

Druhá fáze:

`AvgCheckIfAllInstalled(RecvBuffer)` – Jednoduše porovná seznam přijatých informací o komponentách klienta se seznamem komponent ,které server zná.Pokud je chybějící komponenta politikou serveru nastavená na pravidlo, které ji vyžaduje (Running,Warning) není klient shledán jako bezpečný a je mu navrácena chyba o nenainstalované vyžadovné komponentě.

Implementováno v souboru: \NAP\SHV\SHV\avginfo.cpp (viz. Příloha 2)

4.2.6 Odeslání dat s výsledkem validace

Odesílání dat ke klientovi probíhá stejně, jako odesílání dat klienta k serveru akorát s tím rozdílem, že první zprávou je celkový výsledek validace.

Na začátek odesílaných dat připojíme výsledek validace klienta

```
value.codesVal.count    = 1;
value.codesVal.results = &validationResult;

pSohResponse->
AppendAttribute(sohAttributeTypeComplianceResultCodes,&value);
```

Proměnná `validationResult` je nastavena na hodnotu výsledku validace (viz.validační funkce)

Odešleme všechny zprávy o chybách

Během validace se vector `m_avgresponsvector` plní informacemi o komponentách, které nejsou v souladu s politikou. Postupné procházení vectoru a odesílání zpráv.

```
for ( avg.m_avgresponsvector_it=avg.m_avgresponsvector.begin() ;
avg.m_avgresponsvector_it < avg.m_avgresponsvector.end();
avg.m_avgresponsvector_it++ ){

value.vendorSpecificVal.vendorId = AvgNapSystemID;
value.vendorSpecificVal.size = sizeof(struct AVG_RESPONSE_DATA);
value.vendorSpecificVal.vendorSpecificData =
(PBYTE)&(*avg.m_avgresponsvector_it);


hr = pSohResponse->AppendAttribute(sohAttributeTypeVendorSpecific,&value);

}
```

Data odeslané ze serveru jsou klientem zpracována.V případě, že je některá z komponent nastavená do nevyhovujícího stavu, bude se klient snažit danou komponentu opravit.

Vectro `m_avgresponsvector` obsahuje jako své prvky struktury

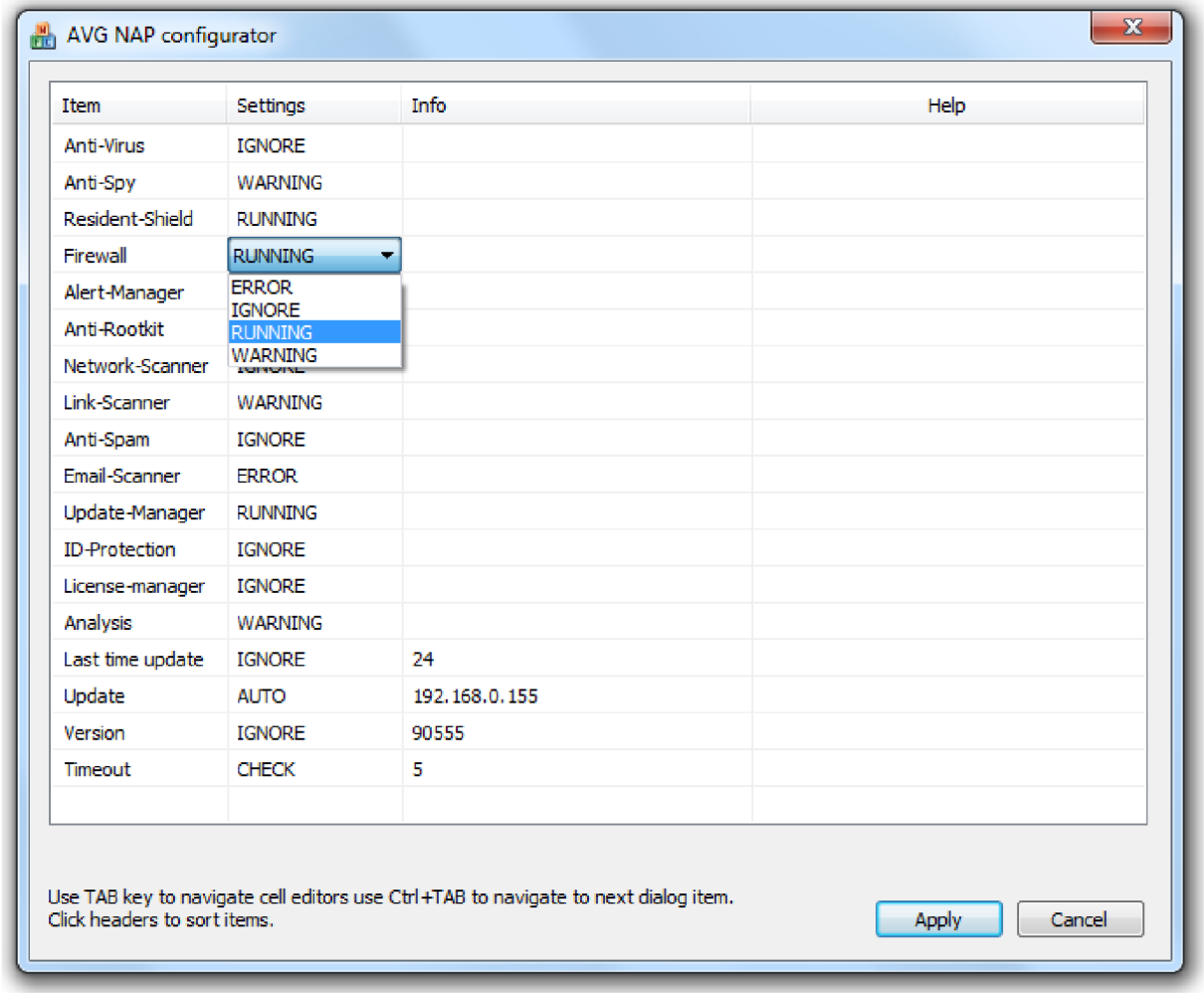
```
struct AVG_RESPONSE_DATA {

            AvgInt32 ComponentID;
            AvgInt32 State;
            char info[30];
};
```

Implementováno v ShvCallback.cpp,*.h (viz. Příloha 2)

4.3 Návrh grafického rozhraní

Grafické rozhraní serveru je navrženo tak, aby jeho ovládání bylo intuitivní, lehce pochopitelné a především se v něm dalo rychle a přehledně nastavit vše co je potřeba. GUI rozhraní je pouze ilustrativní a pouze k testovacím účelům.



4.3.1 Grafické rozhraní SHV validátoru.[12]

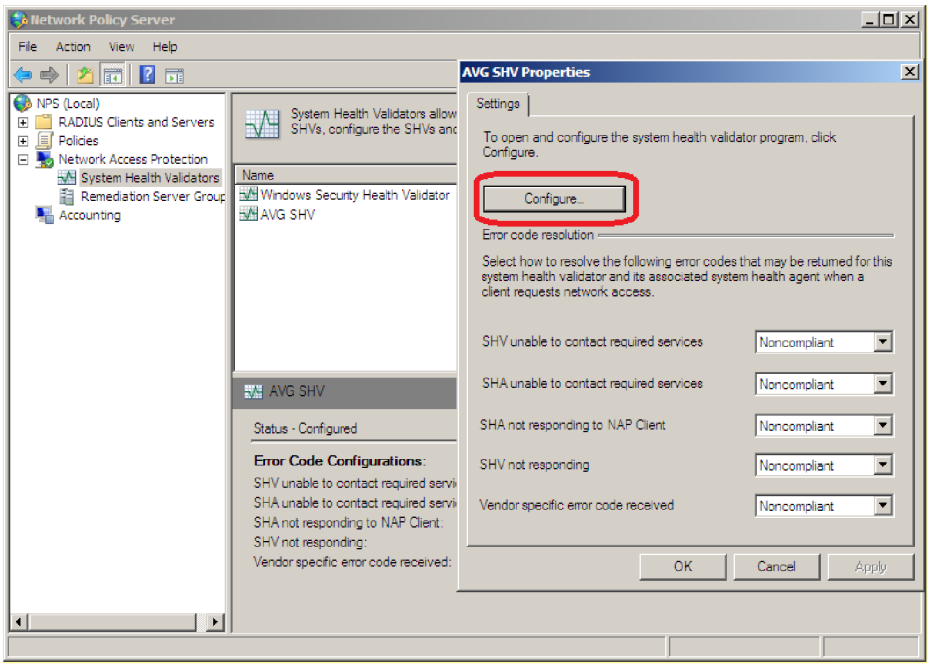
Rozhraní čte a ukládá nastavenou konfiguraci do souboru politik ve formátu XML.

Nejdůležitějším aspektem GUI pro tuto aplikaci je jeho snadná rozšiřitelnost o nové položky. Předpoklad je takový, že s každou novou verzí se budou vyskytovat nové komponenty a položky, které bude třeba implementovat do SHV serveru a GUI tyto položky musí podporovat a umět konfigurovat.

Registrační aplikace GUI

Další částí GUI je registrační aplikace, která se při instalaci zaregistruje v systému a umožňuje GUI rozhraní spouštět z konfiguračního GUI daného SHV v administraci NPS .

[12] Zdroj: vlastní.



4.3.2 Konfigurace SHV validátoru v NPS.[13]

Aplikace se zaregistuje v systému. Pokud administrátor rozklikne nastavení AVG SHV serveru, tak se automaticky spustí hlavní okno GUI.

[13] Zdroj: vlastní.

4.4 Instalační balíček

Instalační balíček obsahuje soubory pro chod SHV validátoru. Nainstaluje do systémového adresáře Shv.dll, Uiinvoker.exe a AVGNAP.exe.
Zároveň hned během instalace zaregistruje Shv.dll komponentu do operačního systému. Rovněž proběhne registrace GUI, které je po instalaci okamžitě zobrazitelné i přes konfiguraci NPS u AVG SHV konfigurace.

Instalační balíček obsahuje všechny potřebné soubory k instalaci komponent NAP na serverové straně. Komponenty klientské strany jsou dodávány s instalačním balíčkem produktu AVG.

5 Závěr

Výsledkem této práce je ucelený přehled o tom, co technologie NAP nabízí, jak této technologie využít a především jakým způsobem technologii dále v produktech implementovat. Čtenář získá představu o rozsahu návrhu a obecný návod, jak technologii implementovat do produktů třetích stran.

Podpora NAP pro antivir AVG vytvořená v této práci, byla úspěšně testována na verzích antivirů 9 a 9,5. Bylo zvažováno nasazení NAP již od verze 10, nicméně se do vydání této práce stále nerozhodlo o verzi antiviru, ve které bude podpora NAP implementována a stále zůstává na pořadníku projektů k realizaci. Projekt rovněž zastal funkci analýzy technologie NAP pro budoucí možná použití v dalších produktech AVG Technologies.

Hlavním přínosem práce je budoucí nasazení technologie NAP do produktů AVG, což bude znamenat značný krok kupředu v prevenci sítí a jejích uživatelů proti škodlivému software a jeho šíření.

Literatura

[1] HOFFMAN, Daniel . *NAP and NAC Security Technologies : The Complete Guide to Network Access Control*. [s.l.] : John Wiley & Sons, Inc, 2008. 288 s. ISBN 978-0-470-23838-7.

[2] DAVIES, Joseph; NORTHRUP, Tony. *Windows Server® 2008 Networking and Network Access Protection (NAP)*. [s.l.] : Microsoft, 2008. 848 s. ISBN 0-7356-2422-4.

[3] KAČMÁŘ, Dalibor. *Programujeme v COM a COM+*. [s.l.] : Computer Press, 2000. 326 s. ISBN 8072263811.

[4] ARLOW, Jim; NEUSTADT, Ila. *UML a unifikovaný proces vývoje aplikací* . [s.l.] : Computer Press, 2003. 387 s. ISBN 80-7226-947-X.

[5] Microsoft. *Technet* [online]. 2011 [cit. 2011-05-11]. Network Access Protection. Dostupné z WWW: <http://technet.microsoft.com/en-us/network/bb545879>.

[6] Microsoft. *Technet* [online]. 2011 [cit. 2011-05-11]. Network Access Protection Design Guide. Dostupné z WWW: <http://technet.microsoft.com/cs-cz/library/dd125338(WS.10).aspx>.

[7] Microsoft. *Network Access Protection Platform Architecture* [online]. 2004. [s.l.] : Microsoft , 2004, 2008 [cit. 2011-05-11]. Dostupné z WWW: <http://download.microsoft.com/download/3/9/f/39ff0ca3-56d1-4d93-af46-98f92134d040/NAPArch.doc>.

[8] Microsoft. *Introduction to Network Access Protection* [online]. 2004. [s.l.] : Microsoft , 2004, 2008 [cit. 2011-05-11]. Dostupné z WWW: <http://download.microsoft.com/download/8/d/9/8d9b3e54-6db7-4955-9e36-58a3f0534933/NAPIntro.doc>.

[9] Microsoft. *MSDN* [online]. 2011 [cit. 2011-05-15]. Service Functions. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/ms685942%28v=VS.85%29.aspx>.

[10] Microsoft. *MSDN* [online]. 2010 [cit. 2011-05-15]. Setting Up a Simple SHA. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/aa369718%28v=vs.85%29.aspx>.

[11] Microsoft. *MSDN* [online]. 2010 [cit. 2011-05-15]. Example SHV. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/aa369152%28v=VS.85%29.aspx>.

[12] Microsoft. *MSDN* [online]. 2010 [cit. 2011-05-15]. SHV Module. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/aa369719%28v=VS.85%29.aspx>.

[13] AVG Technologies. *Avgwiki : internal portal* [online]. 2010 [cit. 2011-05-15]. AVG components. Dostupné z WWW: <http://wiki.portal.avg.cz/avg/components>.

[14] AVG Technologies. *Avgwiki : internal portal* [online]. 2010 [cit. 2011-05-15]. AVG components. Dostupné z WWW: <http://wiki.portal.avg.cz/avg/components/watchdog/>.

[15] AVG Technologies. *Avgwiki : internal portal* [online]. 2010 [cit. 2011-05-15]. AVG components. Dostupné z WWW: <http://wiki.portal.avg.cz/avg/components/config/settings>.

Seznam příloh

Příloha 1. Manuál s návodem na instalaci.
Příloha 2. CD/DVD obsahující zdrojové kódy klientské i serverové strany a instalační balíčky.
Příloha 3. Úpravy SHA a SHV, návrh grafického rozhraní.

Příloha číslo 1.

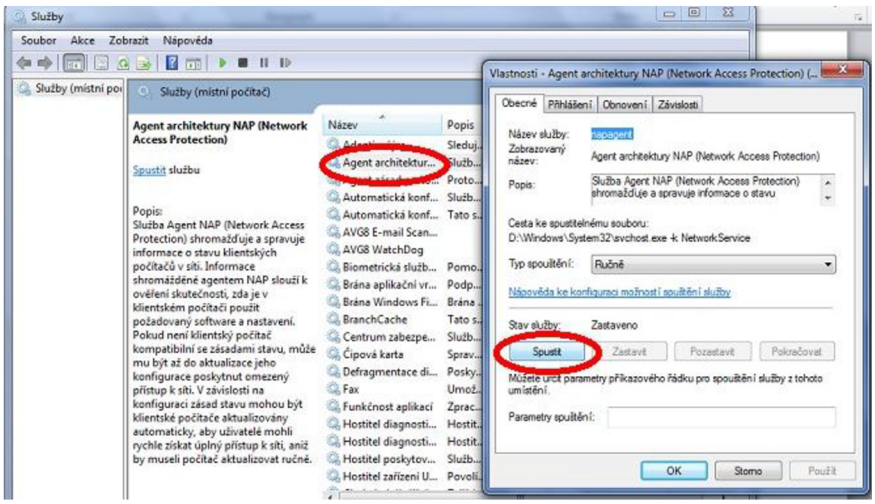
Manuál s návodem na instalaci.

Instalace klientské části projektu (SHA)

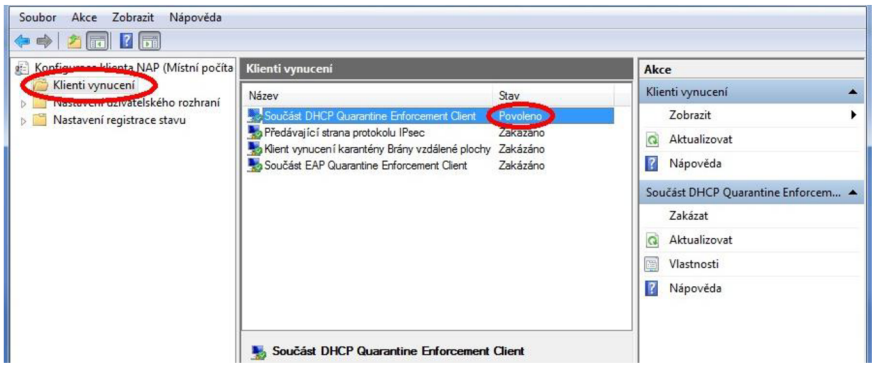
Po kompilaci projektů AvgInfoLib a SHA vznikne soubor **Sha.exe.** Předpokladem pro běh komponenty je Microsoft Windows XP SP3 a novější.

1, Nastavení operačního systému Windows pro podporu služeb NAP.

Nejprve je nutné spustit NAP agenta. Je to služba systému Windows, která je ve výchozím nastavení vypnutá. Ke spuštění služby pomocí GUI Windows stačí otevřít okno správce služeb příkazem spustit **Services.msc** nebo přes správce úloh.



2, Nyní nám v systému běží služba agenta NAP. Dalším krokem je aktivace klienta vynucení EC (Enforcment Client). Příkazem spustit **napclcfg.msc** otevřeme konfigurační okno klientů vynucení.



Pro kontrolu zda je NAP agent správně spuštěn lze otevřít konzolu cmd.exe **s právy administrátora** a použít příkaz ***netsh nap client show state***.

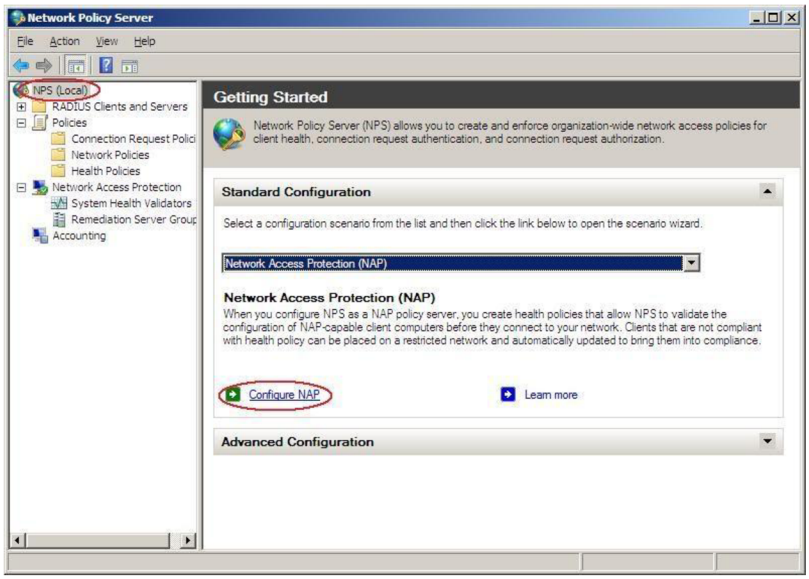
Další informace týkající se instalace a spuštění NAP agenta na klientské stanici:
http://technet.microsoft.com/en-us/library/dd348494%28WS.10%29.aspx

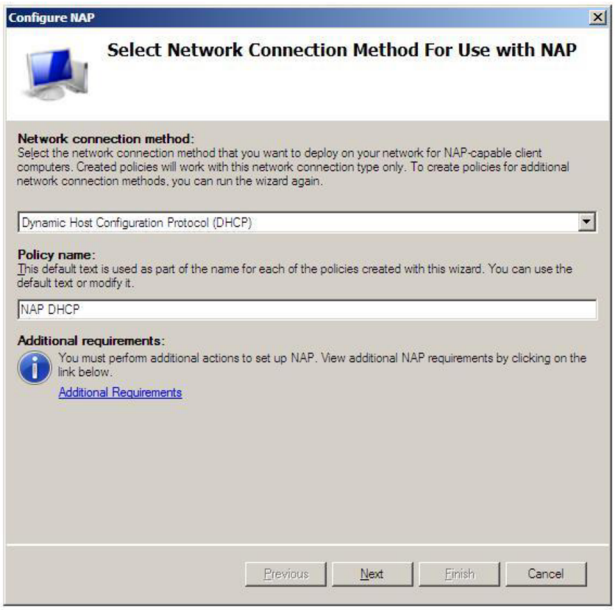
3, Posledním krokem je spuštění samotné komponenty agenta **Sha.exe s právy administrátora!!**
Bez práv administrátora se SHA agent nebude schopen připojit k NAP agentovi.
Samotný SHA agent běží jako služba systému Windows. Najdeme ji tedy spuštěnou po instalaci v service manažeru operačního systému. (services.msc)
Agent Sha.exe musí být umístěn ve složce antiviru AVG. Poté může být spuštěn z příkazového řádku s parametrem `"install".`
Parametr `"install"` je volán pouze jednou při instalaci služby SHA agenta. Služba se po instalaci spouští automaticky s každým startem systému Windows.

Instalace serverové části projektu (SHV)

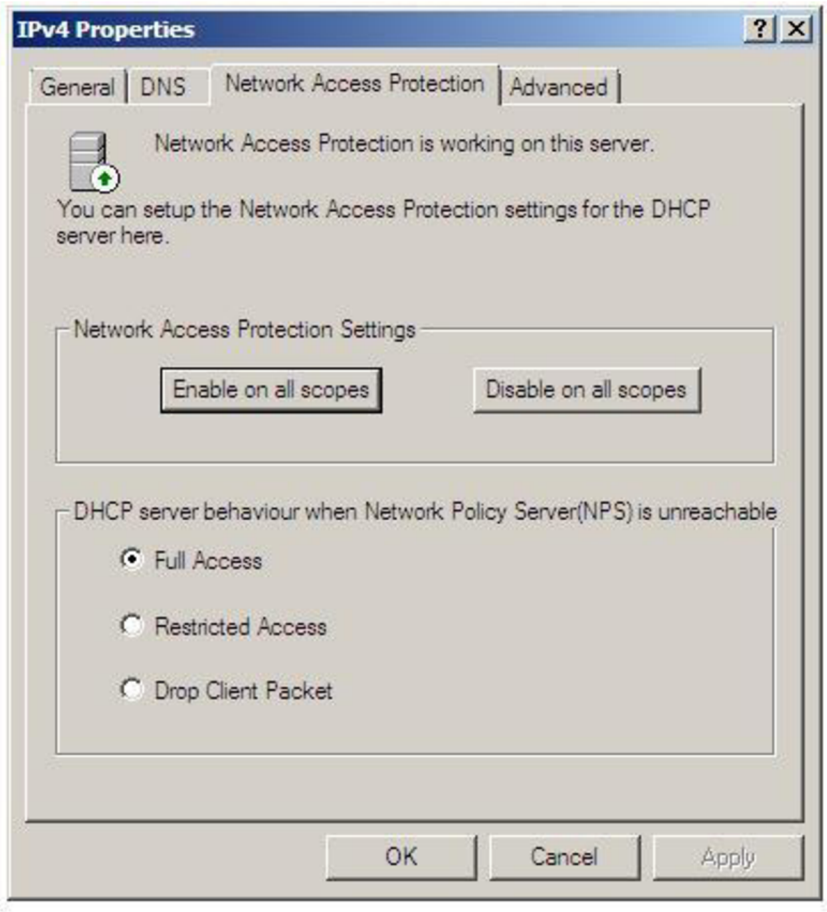
Výsledkem kompilace projektu SHV je knihovna SHV.dll.
Předpokladem pro běh komponenty je Microsoft Windows Server 2008 a novější. Níže uvedená konfigurace je ve spojitosti s klientem vynucení typu **DHCP.**

1, Nastavit statickou IP adresu serveru.(kvůli konfiguraci DHCP serveru) např. 192.168.0.1
2, Otevřít Server Manager a přidání NPS serveru (Add role->NPS)next, next.
3, Otevřít Server Manager a přidání DHCP serveru (Add role->DHCP)next, next.
4, Otevřít DHCP manager, přidat scope (Add scope) (nastavit rozsah přidělování IP adres klientům sítě)
5,NPS manager -> Configure NAP -> DHCP ->NEXT->next->finish

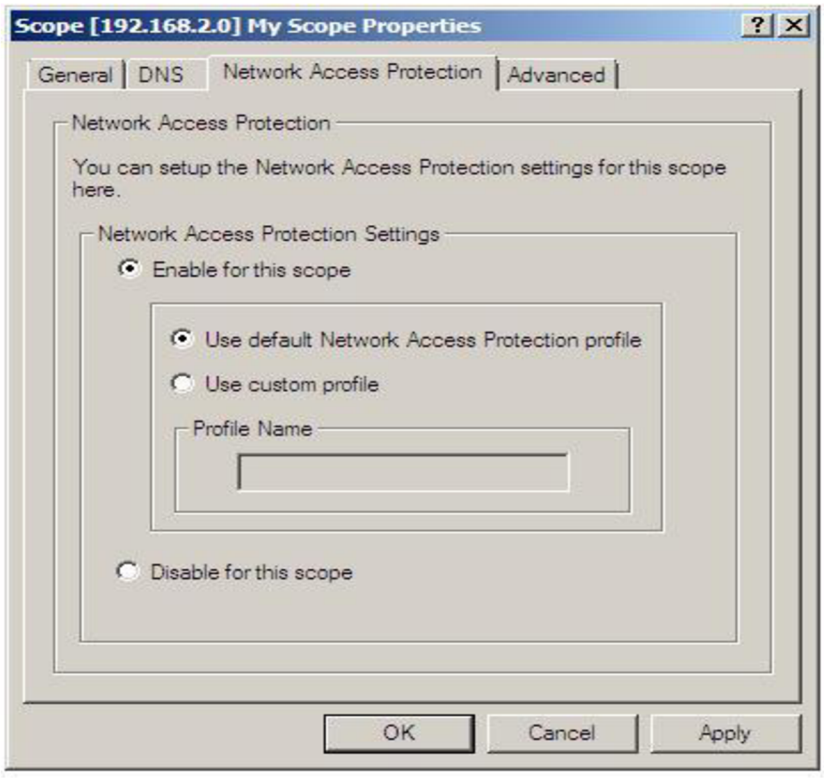




6, DHCP manager ->properties IPv4 ->Enable on All scopes



7, DHCP manager ->properties dané scope (enable for this scope)



Nyní máme server připraven pro nasazení pro použití technologie NAP ve spojení s DHCP serverem. Klient, který se bude pokoušet dostat od našeho serveru IP adresu bude požádán o data ze svých SHA agentů.

Dalším krokem je spuštění samotné komponenty SHV. Komponenta je ve formě DLL knihovny, kterou je třeba do systému zaregistrovat pomocí spuštění programu s parametrem:

Instalace - regsvr32 Shv.dll

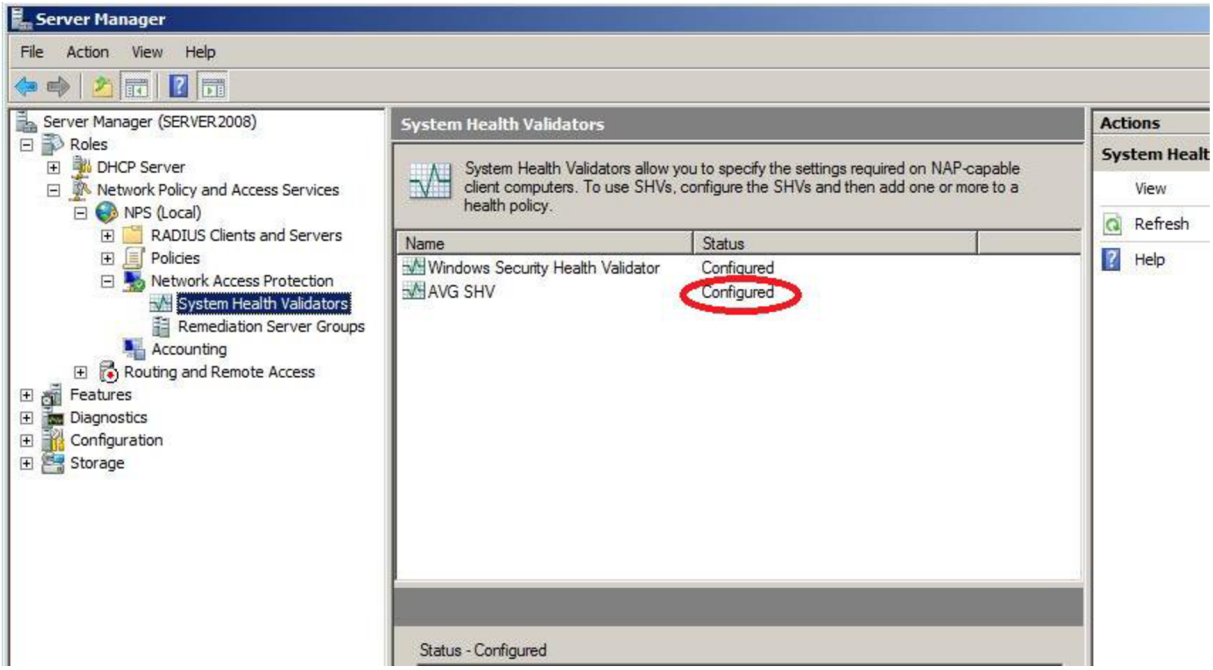
Odinstalace - regsvr32 /u Shv.dll

Příkaz vypíše MessageBox v případě úspěšného i chybného zaregistrování komponenty.

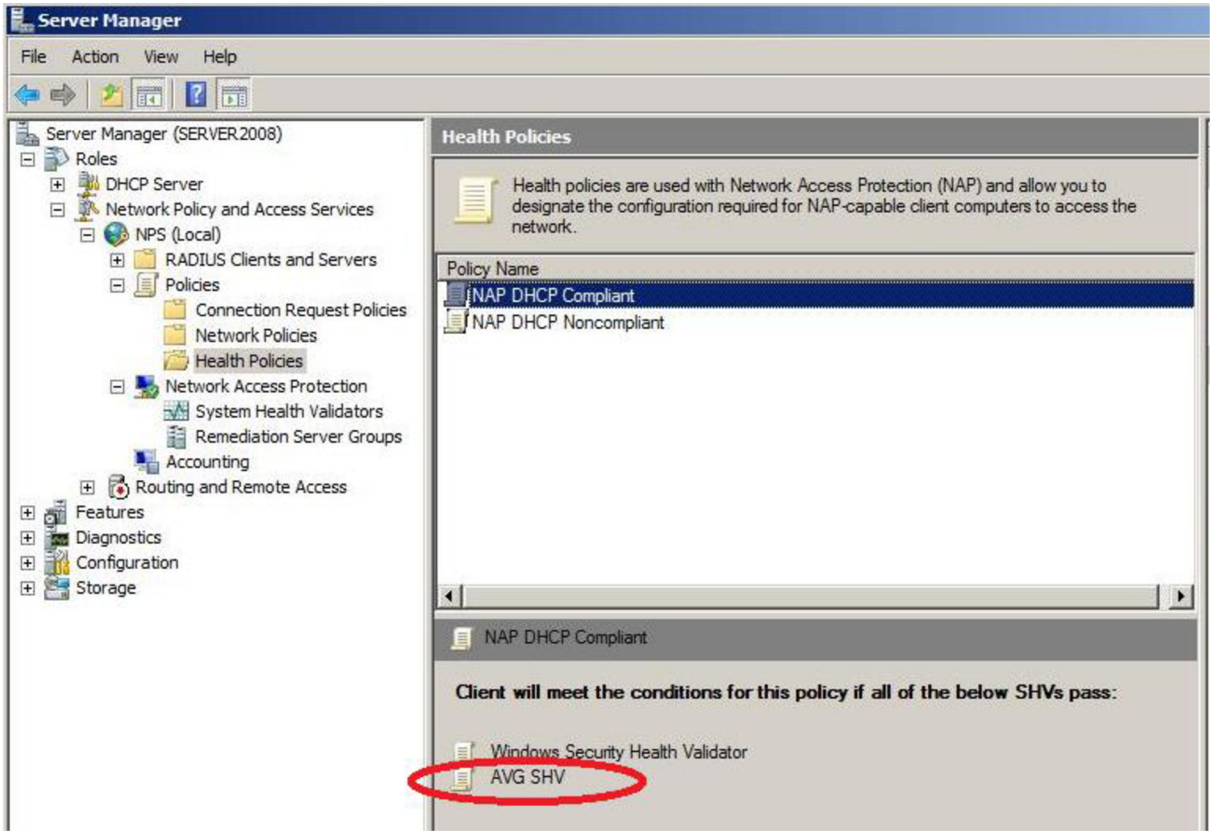
Příkaz regsrv32 pouštíme z příkazového řádku (spustit cmd.exe s právy administrátora!!) v aktuálním umístění knihovny Shv.dll

Nakopírování konfiguračního souboru **AvgShvConfig.xml** do složky **system32** systému Windows.

V nastavení NPS serveru se lze přesvědčit, zda byl AVG SHV zaregistrován.



Zároveň musíme nastavit AVG SHV do politiky serveru. V kolonce policies zvolíme Health Policies a u každé z nich zkontrolujeme vlastnosti. Přidáním (zaškrtnutím) AVG SHV v seznamu zaručíme, že při každé kontrole klienta bude použit rovněž AVG SHV validator.



Registrace SHV grafického rozhraní

Pro registraci pomocí příkazového řádku spuštěného s právy administrátora se dostaneme do složky grafického rozhraní SHV. Spustíme sampleshvui.exe s parametrem regserver nebo unregserver.
Příklad:

sampleshvui.exe /regserver

sampleshvui.exe /unregserver

Běh grafického rozhraní není nutný pro běh serveru. Grafické rozhraní pouze konfiguruje AvgShvConfig.xml konfigurační soubor.

Příloha číslo 3.

Úpravy SHA a SHV, návrh grafického rozhraní

Upgrade SHA agenta

Důvody upgrade :

1. Nové komponenty v antiviru, které je třeba přidat do kontroly NAP
2. Změna komponent
3. Přechod na vyšší verze komponent
4. Změna stavů komponent

Přidání kontroly komponenty do klienta SHA.

Čtení a odeslání stavu komponenty

1. Přidání konstant do souboru ..\NAP\AvgInfoLib\AvgInfo.h
   Přidat do

   ```
   enum EAvgComponents
   struct SAvg_All_ComponentState
   struct SAvg_OnlyStatus_ComponentState
   ```

2. Přidání načtení stavu komponenty do funkce (..\NAP\AvgInfoLib\AvgInfo.cpp)
   `CAvgInfoReader::ReadAllComponentState`

3. Přidání komponenty do odesílaného vectoru (..\NAP\SHA\SHA\EXE\getavginfo.cpp)
   Do funkce
   `AvgErrorCode FillSendVector`

Přidání sledování stavu komponenty

Stačí přidat hlídač v inicializaci třídy pro novou komponentu

`ComponentStateWatcher::ComponentStateWatcher()`

V souboru `..\NAP\AvgInfoLib\ComponentStateWatcher.cpp`

Přijetí zprávy o chybě komponenty

V první řadě je nutné přidat komponentu do funkce `GetAvgInfo::CheckSHVresponse` v souboru

`..\NAP\SHA\SHA\EXE\getavginfo.cpp`

Tato funkce je volána s každou přijatou zprávou od serveru.Přidáme kontrolu chybové zprávy pro novou komponentu podle jejího ID.

Například :

```
if(AVG_DATA->ComponentID==NEWCOMPONENT_KEY_ID){ rekace na chybu viz.dále }
NEWCOMPONENT_KEY_ID - KEY_ID nové komponenty ze souboru wddefs.h
```

Nastavení komponenty do správného stavu

Reakcí na chybu komponenty je nastavení komponenty do správného stavu.

První možností je samotné spuštění komponenty, pokud je vypnutá pomocí komponenty WatchDog. Stačí zavolat SendCommand s parametrem, který bude odpovídat nové komponentě. Parametry zpráv jednotlivých komponent jsou uvedeny v souboru .. \common\protocols\avgwdmsg.h, kde si najdeme příkazy i pro novou komponentu. Pak stačí pouze použít funkci SendCommand.

Příklad pro novou komponentu v souboru `..\NAP\SHA\SHA\EXE\getavginfo.cpp ve funkci GetAvgInfo::CheckSHVresponse` tedy bude následující :

```
if(AVG_DATA->ComponentID==NEWCOMPONENT_KEY_ID){
//Reakce na chybu
   m_componentsSetting.SendCommand(WD_COMMAND_START_ NEWCOMPONENT);
}
```

Třída pro přístup ke komponentě WatchDog implementována v souboru

`\NAP\AvgInfoLib\ComponentStateSetting.cpp`

CONFIG

Dalším nastavením, je nastavení konfigurace komponenty v konfiguračním souboru. Přistup ke konfiguračním souborům je implementován v souboru \NAP\AvgInfoLib\ComponentCfgSetting.cpp. Různé komponenty potřebují různé konfigurační soubory. Většinou pro určitou skupinu komponent stačí nahrát a ukládat do jednoho konfiguračního souboru.Většině komponent stačí `CKEYID_krnl` soubor.Například pro Antivir,antispam,resident-shield a pod.

Pokud nová komponenta vyžaduje svůj konfigurační soubor,musíme načtení přidat do inicializace v `CAvgCfgSetting::Initialize()` v souboru

`\NAP\AvgInfoLib\ComponentCfgSetting.cpp`

Stačí zavolat funkci pro zápis hodnoty do konfigu

Příklad nastavení Resident-shield

```
if(AVG_DATA->ComponentID==RS_KEY_ID){
            //Send command to start RS
m_componentsSetting.SendCommand(WD_COMMAND_START_RS);
m_config.pSetupConfig->putBoolValue(CKEYID_krnl_resident_enabled,true);
m_config.pSetupConfig->ApplyConfig();
}
```

Upgrade SHV validátoru

Důvody upgrade:

1. Nové komponenty v antiviru, které je třeba přidat do kontroly NAP
2. Změna komponent
3. Přechod na vyšší verze komponent

XML reader, XML writer - Přidání komponenty do konfiguračního XML souboru.

Přidání záznamu do struktury `struct ShvXmlConfig`

V souboru XmlConfReader.cpp přidání načtení konfigurace komponenty ze souboru

Například pro komponentu Anti-Spam:

```
//Anti-Spam
	queryNodes((char*)L"//Anti-Spam/*");
	if(!result.empty()){
		if(result.back().compare("running") == 0){
			config->AntispamConf.state=RUNNING_STATE;
		}
		if(result.back().compare("warning") == 0){
			config->AntispamConf.state=WARNING_STATE;
		}
		if(result.back().compare("error") == 0){
			config->AntispamConf.state=ERROR_STATE;
		}
		if(result.back().compare("ignore") == 0){
			config->AntispamConf.state=IGNORE_STATE;
		}
	}
	else{
		config->AntispamConf.state=IGNORE_STATE;
	}
```

Přidání komponenty do XmlConfWriter.cpp (viz. Příloha 2)

```
//Anti-Spam

//Save component state with all atributes
CHK_HR(CreateAndAddElementNode(pXMLDom, L"Anti-Spam", L"\n\t", pRoot,
&pNode));
//Save component atributes
CHK_HR(CreateAndAddElementNode(pXMLDom, L"state", L"\n\t\t", pNode
,&pSubNode));

CHK_HR(CreateAndAddTextNode(pXMLDom, GetStringFromState(config-
>AntispamConf.state), pSubNode));

SAFE_RELEASE(pSubNode);
SAFE_RELEASE(pNode);
```

Přidání kontroly komponenty

`AvgModule::AvgValidateDataVector`

Přidání kontroly přijaté informace od klienta a stavu v jakém by měla komponenta být podle konfiguračního souboru.

Například pro Anti-Spam

```
if(RecvBuffer->compID == ANTISPAM_KEY_ID){

            if(RecvBuffer->component.state < config.AntispamConf.state ){

            UnhealthyState=1;

            AVG_RESPONSE_DATA *AntispamState = new AVG_RESPONSE_DATA;
            AntispamState->ComponentID=ANTISPAM_KEY_ID;
            AntispamState->State=RecvBuffer->component.state;
            m_avgresponsvector.push_back(*AntispamState);
            delete AntispamState;
      }
      }
```

AvgModule::AvgCheckIfAllInstalled

Přidání kontroly nové komponenty do `AvgCheckIfAllInstalled`. Pokud klient neodešle informaci o komponentě,tak se porovná, zda je komponenta nutná pro splnění bezpečnostních politik. Pokud ano, klient bude odmítnut.

```
if(config.AntispamConf.state != ERROR_STATE && config.AntispamConf.state
!= IGNORE_STATE){
            if(RecvBuffer_it->compID ==  ANTISPAM_KEY_ID){
                  is_installed->AntispamState=true;
            }
            else{
                  if(is_installed->AntispamState!=true)
                  {
                        is_installed->AntispamState=false;
                  }
            }
      }
      else{
            is_installed->AntispamState=true;
      }
```

Návrh nového grafického rozhraní.

Network Access Protection configuration tool for AVG.

Návrh webového editoru konfiguračního souboru. (NAPct.png)