Windows Update 및 Automatic Updates의 설계 오류 분석 및 해결 방안

김윤주* · 윤영태* · 강성문**

요 약

마이크로소프트에서 보안패치 적용을 쉽고 빠르게 할 수 있도록 지원하는 윈도우즈 업데이트(Windows Update)와 자동 업데이트(Automatic Updates)의 적용 대상 보안패치 검색 모듈에서 설계상의 오류가 발견되었고, WFP(Windows File Protection)에 대한 우회 방법이 공개되었다. 이로 인해 취약점을 그대로 유지하면서 보안패치 검색이 되지 않도록 하는 보안패치 가장 공격이 가능함을 입증하고, 위협 시나리오를 구성한다. 그리고 보안패치 검색 모듈의 설계상 오류를 해결하기 위한 방안을 제안한다.

Analysis of Design Error in Windows Update and Automatic Updates, and the Solutions

Yun-Ju Kim* · Young-Tae Yun* · Sung-Moon Kang**

ABSTRACT

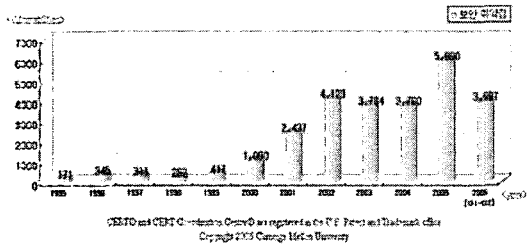
It discovered a design error from the module to search required installing security patch of Windows Update and Automatic Updates that Microsoft supports it to install security patch easily and quickly. It explains and tests security patch-disguise attack by this error. Security patch-disguise attack is to maintains a vulnerability and to be not searched the security patch simultaneously. Also it composes an attack scenario. Is like that, it proposes the method which solves an design error of the module to search required installing security patch.

Key words : Security Patch, Windows Update, Automatic Updates, Vulnerability, Malware

* 국가보안기술연구소
** 고려대학교

1. 서 론

보안취약점을 이용하여 권한 상승, 임의의 코드 실행, 서비스 거부 등의 공격을 수행하는 익스플로잇이 개발되어 공개됨에 따라 수많은 공격 시도가 이루어지고 있다. CERT/CC(Computer Emergency Response Team Coordination Center)는 1995년부터 2006년 상반기까지 소프트웨어 결함에 의한 보안 취약점은 26,713건이라고 보고하였으며 그 변화추이는 (그림 1)과 같이 꾸준히 증가하고 있다[1].

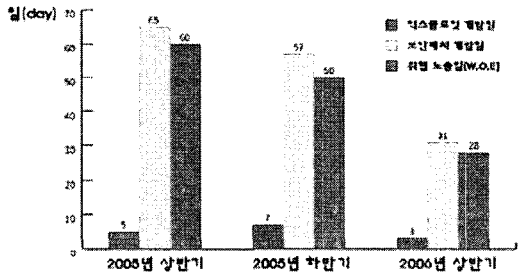


(그림 1) 보안취약점 보고 현황
(1995~2006 상반기)

소프트웨어 취약점을 악용한 공격에 대응하기 위해서는 해당업체에서 제공하는 보안패치를 빠르게 적용하는 것이 필요하다. 마이크로소프트사에서도 이를 위해 홈페이지를 통한 윈도우즈 업데이트(Windows Update - 이하 *WU*)와 윈도우즈 운영체제의 서비스로 동작하는 자동 업데이트(Automatic Updates - 이하 *AU*) 기능을 제공하고 있다.

2006년 상반기 시만텍의 인터넷 보안 위협 보고서 제10호에 의하면, (그림 2)와 같이 패치 개발 기간의 단축으로 W.O.E.가 지속적으로 감소하고 있음을 알 수 있다[2].

W.O.E.는 Window Of Exposure의 약자로, 발견된 보안 취약점에 대해 해커는 익스플로잇을 개발하고, 해당 업체는 취약점을 수정하는 보안패치를 개발하는데, 이때 익스플로잇이 개발된 시점부터 수정 패치가 배포되기 전까지인 위협 노출 기간을 의미한다. 특히 마이크로소프트는 패치 개발 시간을 평균 13일로 단축시켜 W.O.E.가 다른 업체보다 짧아지고 있다[2].



(그림 2) W.O.E. 통계(2005~2006 상반기)

그러나 마이크로소프트의 노력으로 보안패치가 신속하게 적용되고 W.O.E.가 단축되더라도, *WU*나 *AU*가 보안패치 적용여부를 정확하게 판단할 수 없다면 최신 업데이트를 모두 적용하였기 때문에 안전하다고 믿고 있는 사용자의 시스템에는 여전히 취약점이 존재하게 된다.

본 논문의 2장에서 *WU*, *AU*의 보안패치 검색 방법을 살펴보고, 3장에서 *WU*나 *AU*의 잘못된 설계로 인해 적용되지 않은 패치가 적용된 것처럼 인식하도록 하는 것을 입증한다. 4장에서 이를 이용한 위협 시나리오를 설명하고, 5장에 대한 해결 방안을 제안하며, 6장에서 결론을 맺는다.

2. 마이크로소프트 *WU*와 *AU*의 보안패치 검색 방법

*WU*는 온라인상에서 홈페이지 형태로, *AU*는 윈도우즈의 서비스로 동작하는 마이크로소프트의 패치 관리 도구로, 사용자의 운영체제를 최신의 상태로 유지할 수 있도록 현재 패치 상태를 검사하고, 필요한 패치가 있는 경우 설치를 지원한다[3, 4]. 특히 *AU*는 마이크로소프트에서 윈도우즈 2000이

후의 모든 버전에 대해서 관리자로 로그인하면 사용 중인 컴퓨터에 설치할 필요가 있는 중요 업데이트가 있을 경우 해당 사실을 자동으로 알려주는 모듈로서, 부가적으로 컴퓨터에 필요한 보안패치를 검색할 주기를 지정해주거나, 백그라운드에서 자동으로 보안패치를 윈도우즈 업데이트 사이트로부터 다운로드 받아서 트레이아이콘 및 툴 팁 등을 이용하여 사용자에게 보안패치를 설치할 것을 권고해 주며, 심지어 사용자의 환경설정에 따라서는 자동으로 설치해주는 기능까지 포함하고 있다.

해당 시스템에 적용대상이 되는 보안패치를 검색하는 방법은 *WU*와 *AU*가 동일하다. (그림 3)의 예와 같은 item.xml 파일의 데이터를 이용하여 각 보안 패치가 해당 시스템에 설치되어 있는지 여부와 설치 제외 대상인지의 여부를 판단하여 설치되지 않았고, 제외 대상이 아닌 경우 설치가 필요한 패치로 결정한다.

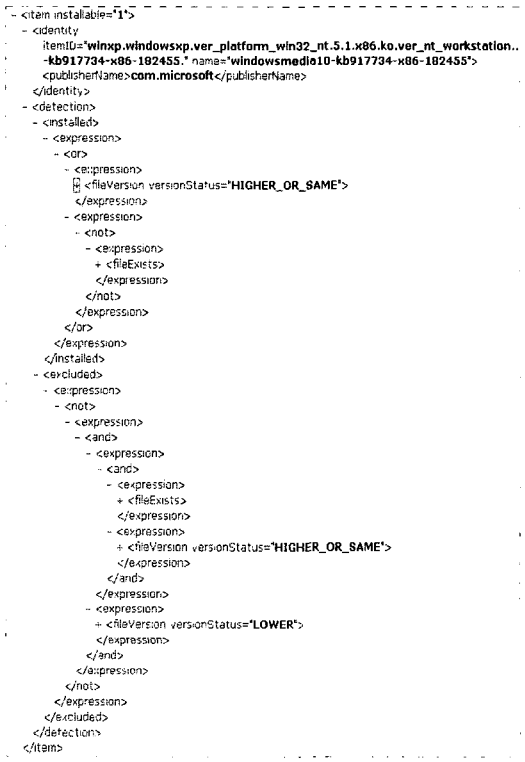
item.xml 파일을 이루는 태그의 구성과 각 태그에 대한 설명을 살펴본다.

2.1 Tag : 〈item〉

각각의 보안패치에 대한 정보가 item 태그의 하위에 위치한다. 즉, 검토하는 보안패치의 개수와 item 태그의 개수가 같다.

2.2 Tag : 〈identity〉

각 보안패치를 구분하는 정보를 포함하는 태그로, itemID와 name 태그로 구성된다. itemID는 [Provider ID]. [Product ID]. [Platform Name]. [Major Version]. [Minor version]. [Processor Architecture]. [Locale]. [Product Type]. [Product Suite]. [Build Number]. [ServicePack Major Version]. [ServicePack Minor Version]. [Publisher Name]. [Patch Name]으로 이루어지고, name은 사람이 식별하기 쉽도록 붙여진 이름이다.



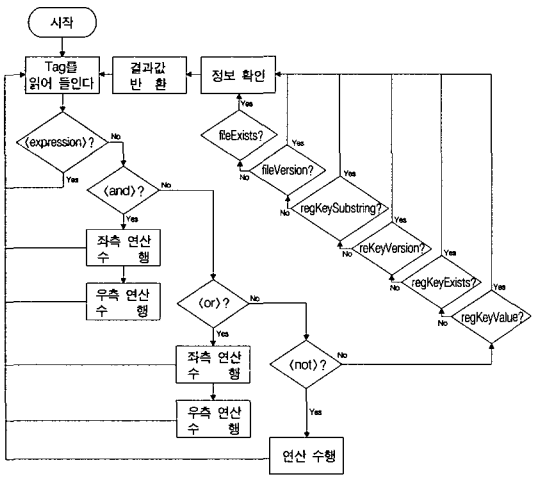
(그림 3) item.xml 파일의 예제

2.3 Tag : 〈detection〉

detection 태그의 정보로 설치 여부와 제외 대상 여부가 결정된다. installed 태그와 excluded 태그로 이루어지는데 installed 태그의 정보로 설치 여부를

〈표 1〉 detection 태그의 하위 태그 종류

태 그	설 명
regKeyValue	주어진 레지스트리 경로에 주어진 키의 값 확인
regKeyExists	주어진 레지스트리 경로에 주어진 키의 존재 여부 확인
regKeyVersion	주어진 레지스트리 경로에 주어진 키의 버전 확인
regKeySubstring	주어진 레지스트리 경로에 주어진 문자열의 포함 여부 확인
fileVersion	주어진 경로에 파일의 버전 확인
fileExists	주어진 경로에 주어진 파일의 존재 여부 확인



(그림 4) *WU*와 *AU*의 검색 순서도

판단할 수 있고, excluded 태그의 정보로 제외 대상 여부를 판단할 수 있다. installed 태그와 excluded 태그의 정보는 동일한 형태로 구성되며, <표 1>과 같은 태그들로 구성된다. 또한 <표 1>의 각 태그들의 하위 태그는 <표 2>와 같다.

이러한 정보들을 이용하여 *WU*와 *AU*는 (그림 4)의 순서도와 같은 연산을 통해 패치가 설치된 시스템, 적용 제외 대상 시스템, 패치 설치가 필요한 시스템을 구분한다.

3. *WU*와 *AU*의 설계 오류

*WU*와 *AU*가 설치 대상 패치를 검색하는 방법에

〈표 2〉 detection 태그의 각 하위 태그 구성

태 그	엔트리	설 명
regKeyValue	key	레지스트리 키 예) HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion
	entry	레지스트리 엔트리 예) CurrentBuildNumber
	value	레지스트리 값 예) 2195
regKeyExists	key	레지스트리 키 예) HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
	entry	레지스트리 엔트리. 필수는 아님. 예) avserve.exe
regKeyVersion	versionStatus	버전의 상태 1. HIGHER_OR_SAME : 레지스트리에 저장된 버전이 주어진 버전보다 높거나 같을 때 TRUE임. 2. LOWER_OR_SAME : 레지스트리에 저장된 버전이 주어진 버전보다 낮거나 같을 때 TRUE임. 3. SAME : 레지스트리에 저장된 버전과 주어진 버전이 같을 때 TRUE임. 4. LOWER : 레지스트리에 저장된 버전이 주어진 버전보다 낮을 때 TRUE임. 5. HIGHER : 레지스트리에 저장된 버전이 주어진 버전보다 높을 때 TRUE임.
	key	레지스트리 키 예) HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer
	entry	레지스트리 엔트리 예) Version
	version	레지스트리 값. 필수는 아님 예) 5.0.3315.0
regKeySubstring	key	레지스트리 키 예) HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings
	entry	레지스트리 엔트리 예) MinorVersion
	value	레지스트리 값. 문자열임. 예) Q837009
fileVersion	versionStatus	버전의 상태. 예)HIGHER_OR_SAME
	name	파일 이름 예) Inetcomm.dll
	path	파일 경로 예) %CSIDL_SYSTEM%
	version	파일 버전 예) 5.50.4939.300
fileExists	name	파일 이름 예) wmpcore.dll
	path	파일 경로 예) %CSIDL_SYSTEM%

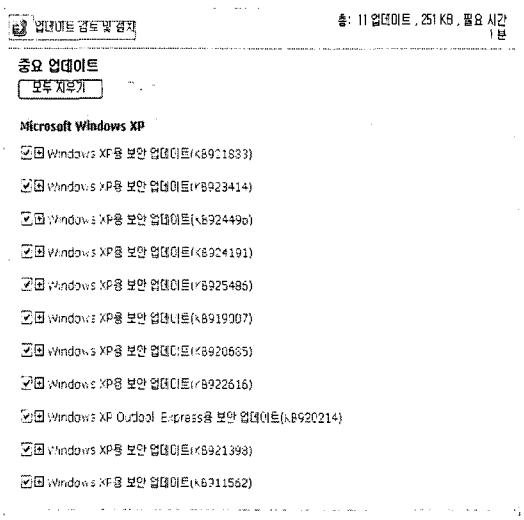
있어서 installed의 조건을 만족하거나 excluded의 조건을 만족하면 해당 패치는 설치 대상에서 제외되므로, 특정 패치가 설치되지 않았더라도 검색과정에서 해당 패치를 설치 대상에서 제외하도록 임의로 조작하는 것이 가능하다.

실험을 통해 조건을 만족시켜 *WU*와 *AU*가 패치를 검색하지 못하는 것을 확인하였다. 실험 방법은 다음과 같이 2가지로 나눌 수 있다.

① 보안패치가 설치되었거나 배제 대상인 것으로 인식되도록 하기 위해서, 관련된 레지스트리의 값을 생성 또는 수정한다.

② 보안패치가 설치되었거나 배제 대상인 것으로 인식되도록 하기 위해서, 관련된 파일들을 임의로 생성 또는 수정한다.

① 방법은 윈도우즈에서 관리자 권한을 가지고 있는 사용자들은 누구나 레지스트리를 쉽게 조작할 수 있기 때문에 가능하다. 그리고 ② 방법을 방지하기 위해 윈도우즈의 시스템 파일들은 Windows File Protection(WFP)[5]을 통해서 보호받고 있지만, WFP를 우회할 수 있는 방법이 여러 가지 공개되어 있기 때문에 ② 방법도 가능하다[6].

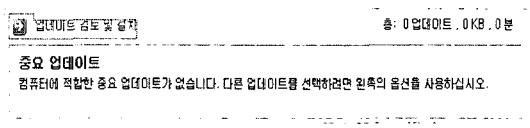


(그림 5) 11개의 보안패치를 검색한 *WU*

취약점을 검증하기 위해 임의의 보안패치 11개를 Windows XP Professional SP2 한글버전에서 테스트하였다. (그림 5)는 *WU*가 11개의 보안패치를 설치해야 하는 시스템으로 판단한 결과를 보여주고 있다. 그러나 <표 3>과 같이 11개의 보안패치가 수정하는 파일에 대해 패치 적용전 버전 파일의 버전을 수정하여 시스템에 복사하고, 다시 *WU*를 실행시켰을 때는 (그림 6)과 같이 적용해야 할 보안패치를 전혀 검색하지 못하였다.

<표 3> 테스트 대상 보안패치와 관련된 조작할 파일 및 버전

925486	vgx.dll	6.00.2900.2180 → 6.0.2900.2997
924496	shdocvw.dll	6.00.2900.2937 → 6.0.2900.2987
924191	msxml3.dll	8.70.1104.0 → 8.70.1113.0
923414	srv.sys	5.1.2600.2893 → 5.1.2600.2974
922616	hhctrl.ocx	5.2.3790.2453 → 5.2.3790.2744
921883	netapi32.dll	5.1.2600.2180 → 5.1.2600.2952
921398	shell32.dll	6.0.2900.2869 → 6.0.2900.2951
920685	ciodm.dll	5.1.2600.2180 → 5.1.2600.2935
	query.dll	5.1.2408.1 → 5.1.2600.2935
920214	inetcomm.dll	6.00.2900.2869 → 6.0.2900.2962
929007	rmcast.sys	5.1.2600.0 → 5.1.2600.2951
911562	msadco.dll	2.81.1117.0 → 2.81.1124.0



(그림 6) 파일 버전 조작 후, 설치할 보안패치를 검색하지 못한 WU

테스트 결과 11개의 보안패치 모두 파일의 정보만 수정하였을 뿐 레지스트리 정보는 수정하지 않았음에도 *WU*와 *AU*가 설치할 보안패치를 모두 검색하지 못하였다. 뿐만 아니라 *WU*와 *AU*의 보

안패치 검색 정보 중 fileVersion 태그의 version Status 정보는 거의 HIGHER_OR_SAME이기 때문에 보안패치가 배포되기 전에도 현재 버전보다 적당히 높은 버전으로 추측하여 수정하여도 이 공격이 성공할 수 있는 확률이 매우 높다.

이와 같이 *WU*와 *AU*가 시스템에 필요한 보안패치를 검색하지 못하도록 하여 취약한 상태를 계속 유지시키는 공격을 "*보안패치 가장 공격*"이라고 정의한다.

4. 위협 시나리오 및 실험

일반적인 악성코드 감염으로 인한 위협의 시나리오는 다음과 같다.

- 대상 시스템에 침입
- 대상 시스템에 악성코드를 상주시켜, 시스템 장악
- 악성코드의 패턴이 안티바이러스에 등록되어 악성코드 탐지, 제거

그러나 악성코드의 패턴이 안티바이러스에 등록되기 전에 *보안패치 가장 공격*으로 원격에서 공격이 가능한 취약점을 수정하는 보안패치를 설치 이전의 상태가 되도록 하면, 악성코드가 탐지, 제거되더라도 공격자가 그 시스템에 다시 침입할 수 있기 때문에 여전히 시스템을 장악할 수 있게 된다.

이러한 방법으로 같은 취약점을 갖는 시스템들이 대량으로 존재하게 하는 것은 네트워크 마비를 목적으로 한 서비스 거부 공격이나, 봇넷(Bot Net) 형성에 이용될 수 있다.

실험은 원격에서 취약점을 이용한 침입이 가능한 MS06-040 취약점을 보완하는 921883 보안패치를 이용하며[7, 8], 익스플로잇 실험은 Metasploit을 활용하여 수행하였다[9]. 실험을 위해 보안패치에 의해 변경되는 파일 정보와 레지스트리 정보를 추출하는 도구(이하 GetPatchInfo)를 구현하였고, PE 형식의 파일 버전을 수정하는 공개된 도구(이하 Resource Editor)를 사용하며[10], WFP를 우회하여 시스템 파일을 복사하는 도구(이하 WFPcopy)를 이용하였다[6]. 실험 환경 운영체제는 MS06-040 취약점에 영향을 받는 Windows 2000 Professional SP4 한국어 버전이다[8].

4.1 보안패치 정보 획득

GetPatchInfo를 이용하여 보안패치 921883은 %System%\netapi32.dll 파일을 수정하고, HKLM\SOFTWARE\Microsoft\Windows NT\Current Version\Winlogon의 Buffer Policy Reads의 값을 생성한 것을 확인하였다. 그 외의 수정된 정보는 보안패치를 적용한 후 제거하기 위한 정보이거나, 적용되었음을 나타내는 레지스트리 정보이다.

이 정보는 *WU*나 *AU*가 적용대상 보안패치를 검색하는 정보에 포함되지 않으므로 실험에 필요하지 않은 정보이다. netapi32.dll은 5.0.2195.6949 버전에서 5.0.2195.7105 버전으로 수정되었고, Buffer Policy Reads의 값은 0x1로 생성되었다. 파일 수정만으로 공격이 가능한 것을 확인하였으므로, 레지스트리 정보도 실험에 사용하지 않는다.

4.2 파일 버전 수정

보안패치가 적용되기 전의 파일인 5.0.2195.6949 버전의 netapi32.dll 파일의 버전을 5.0.2195.7105로 Resource Editor를 이용하여 수정한다.

4.3 시험용 악성코드

WFPcopy 도구를 이용하여 버전이 수정된 netapi32.dll 파일을 시스템 폴더에 복사하는 코드가 포함된 악성코드를 준비한다.

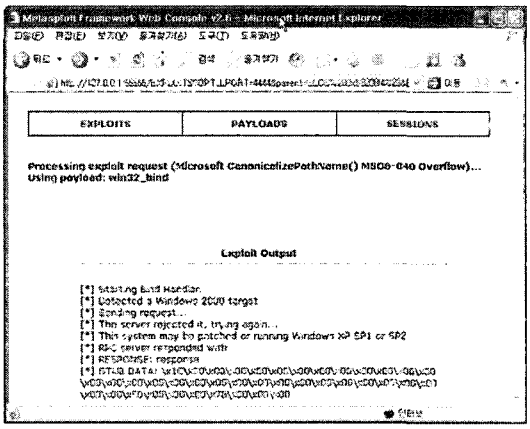
4.4 대상 시스템 침입 및 악성코드 실행

대상 시스템에 침입하는 방법은 취약점이 존재한다면 취약점을 이용하거나, 이메일, P2P, 메신저

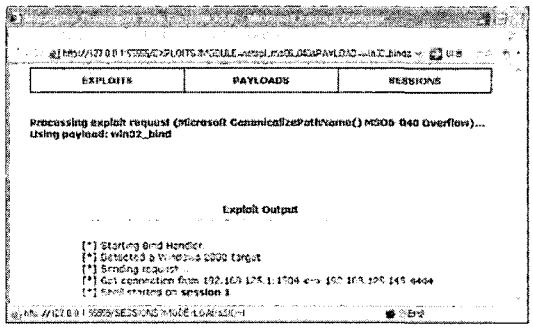
등을 이용하여 사용자의 실수를 유발하는 등과 같이 여러 가지가 있을 수 있다. 어떠한 경로이든 침입했다고 가정하고 대상 시스템에서 시험용 악성코드를 실행시켜, 수정된 netapi32.dll을 복사한다.

4.5 대상 시스템에 MS06-040 취약점을 이용하여 다시 침입

대상 시스템에서 사용하는 안티 바이러스 제품에 의해서 악성코드가 탐지, 제거되었다고 가정하고 다시 대상시스템에 침입할 때 MS06-040 취약점이 여전히 존재할 수밖에 없으므로 첫 번째 침입 과정보다 훨씬 수월하게 침입이 가능하다.



(그림 7) 921883 패치가 적용된 시스템에 시도한 공격의 실패 화면



(그림 8) netapi32.dll의 버전을 조작한 시스템에 시도한 공격의 성공 화면

(그림 7)은 대상 시스템에 보안패치 921883이 적용되어 취약점을 이용하여 쉘을 획득하는 공격에 실패한 화면이고, (그림 8)은 *보안패치 가장 공격*으로 *WU*와 *AU*가 보안패치가 설치된 것처럼 판단하지만 취약점을 이용하여 공격에 성공한 화면이다.

5. 해결 방안

취약점 테스트 결과와 같이 파일의 버전을 수정하는 것만으로 보안패치 가장 공격이 가능하였다. 우선 해당 취약점에 관련된 문제점은 다음과 같다.

- 파일 버전 정보 수정이 가능함에 따라 조작 파일을 생성할 수 있다.
- WFP 우회가 가능하여 조작된 파일로 시스템 파일을 변경할 수 있다.
- *WU*와 *AU*가 정상 파일과 조작된 파일을 구분할 수 없는 문제점이 있다.
- *WU*와 *AU*가 파일 버전 비교시 HIGHER_OR_SAME 상태를 기준으로 하기 때문에 보안패치 배포 전에도 공격이 가능하다.

이를 해결하기 위해서는 WFP 기능을 강화하거나, 보안패치 검색시 fileHash 태그를 추가하여 hash 값을 비교하며, *WU*, *AU*가 보안패치 검색을 위한 정보가 많아질 수 있더라도 파일 버전은 SAME 상태를 기준으로 비교하는 것이 필요하다.

5.1 WFP 기능 강화

어떠한 방법으로도 WFP 기능을 우회할 수 없다면 시스템 파일을 교체할 수 없으므로 *보안패치 가장 공격*은 이루어질 수 없다.

5.2 hash 태그 추가

보안패치 가장 공격에 대응하기 위해서 WU나

AU는 패치를 설치한 후의 파일과 최신 버전을 수정한 패치 설치 전 파일을 구분할 수 있어야 한다. 이를 구분하는 정보로 파일 전체에 대한 Hash 값을 이용하고자 한다.

예로 <표 4>에서 보안패치 921883과 관련된 파일의 해쉬값(MD5)을 비교해 보았고, 모두 다른 값을 갖는 것을 확인하였다.

〈표 4〉 보안패치 921883 관련 파일의 Hash 값

구 분	Hash Value
natapi32.dll (패치 설치전 파일)	d5cca2350b32ce893aeb3e3e4f0471e2
natapi32.dll (패치 설치전 파일의 버전 조작 파일)	d761f53f0117f4e0d8ab2dffef65f93c
natapi32.dll (패치 설치후 파일)	2f6f5280feba7848509a28721d6cd695

그래서 fileVersion 태그 하위에 name 태그, path 태그, version 태그와 더불어 hash 태그를 추가하여 (그림 9)와 같이 기존에 버전만 비교하던 것을 해쉬값도 비교하는 것으로 설계를 변경한다. 변경한 예는 (그림 10)과 같다.

```
- <fileVersion versionStatus="HIGHER_OR_SAME">
  - <filePath name="netapi32.dll">
      <path>%CSIDL_SYSTEM%</path>
    </filePath>
    <version>5.1.2600.2952</version>
  </fileVersion>
```

(그림 9) 현재 보안패치 검색 정보

```
- <fileVersion versionStatus="SAME">
  - <filePath name="netapi32.dll">
      <path>%CSIDL_SYSTEM%</path>
    </filePath>
    <version>5.1.2600.2952</version>
    <hash>2f6f5280feba7848509a28721d6cd695</hash>
  </fileVersion>
```

(그림 10) Hash 값 비교를 추가한 보안패치 검색 정보

6. 결 론

본 논문에서는 Windows Update와 Automatic Updates의 보안패치 검색 과정에서의 설계상 취약점을 발견하고 *보안패치 가장 공격*을 입증하였으며, 취약점에 대한 해결방안을 제시하였다. 이 취약점은 윈도우즈 운영체제를 사용하는 사용자에게 혼란을 야기할 수 있고, 공격자는 한번 장악한 시스템을 계속 제어할 확률을 높이며, DDoS 공격이나 BotNet 형성에 효율성을 증대시킨다. 제시한 해결방안인 WFP 기능 강화와 *WU*, *AU*의 보안패치 검색 정보에서 해쉬값 비교가 조속히 윈도우즈에 적용되어 *보안패치 가장 공격*으로부터 안전한 시스템을 사용할 수 있도록 개선이 필요하다.

참 고 문 헌

[1] CERT/Coordination Center, "CERT/CC Statistics 1988-2006 : Vulnerabilities reported", http://www.cert.org/stats/cert_stats.html, July 2006.

[2] Dean Turner et al., "Symantec Internet Security Threat Report, Trands for January 06-June 06", Symantec, Volume X, September 2006.

[3] Microsoft Corporation, "Deploying Updates with Windows Update and Automatic Updates", http://www.microsoft.com/technet/security/smallbusiness/topics/patchmanagement/dep_patches_wu_au.mspx.

[4] Microsoft Corporation, "Windows Update", http://windowsupdate.microsoft.com/, 2006.

[5] Microsoft Corporation, "Windows File Protection", http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/system_file_protection.mspx?mfr=true.

[6] Jeremy Collake, Bitsum Technologies, "Hacking Windows File Protection", http://www.bitsum com/aboutwfp.asp.

[7] Microsoft Security Bulletin MS06-040, "Vulnerability in Server Service Could Allow Remote Code Execution(921883)", http://www.microsoft com/technet/security/bulletin/ms06-040. mspx, August 2006.

[8] SecurityFocus, "Microsoft Windows Server Service Remote Buffer Overflow Vulnerability", http://www.securityfocus.com/bid/19409/exploit, August 2006.

[9] Metasploit, http://www.metasploit.com.

[10] XN Resource Editor, "http://www.wilsoncdemon.co.uk/d10resourceeditor.htm".

김 윤 주

2003년 고려대학교 전자및정보공학부(공학사)
2005년 고려대학교 정보보호대학원(공학석사)
2005년~현재 국가보안기술연구소 연구원

윤 영 태

1995년 충남대학교 컴퓨터과학과(이학사)
1997년 현대전자 정보시스템 사업본부
1999년 충남대학교 컴퓨터과학과(이학석사)
1999년~현재 국가보안기술연구소 선임연구원
2006년 충남대학교 컴퓨터과학과(이학박사)

강 성 문

1985년 한남대학교 전자계산학과(이학사)
1990년 성균관대학교 컴퓨터감사학과(이학석사)
2004년 고려대학교 정보보호대학원 박사과정
현재 국방 정보전 대응 센터장