

[TOOL] Beta – Multi-Format Shellcode Encoding Tool

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2005-07/0056.html>

From: SecuriTeam (*support_at_securiteam.com*)

Date: 07/21/05

To: list@securiteam.com

Date: 21 Jul 2005 13:43:44 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

Beta – Multi-Format Shellcode Encoding Tool

SUMMARY

DETAILS

Source Code:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
/*
```

```
    ,sSSs, Beta v1.0.
```

```
    dS\" dP Multi-format shellcode encoding tool.
```

```
    .SP dSS\" Copyright (C) 2003 by Berend-Jan Wever
```

```
    dS' Sb <skylined@edup.tudelft.nl>
```

```
    .SP dSSP' Encodes shellcode to a variety of formats.
```

```
    _ iS: _____
```

This program is free software; you can redistribute it and/or modify it under

the terms of the GNU General Public License version 2, 1991 as published by

Securiteam: [TOOL] Beta – Multi-Format Shellcode Encoding Tool

the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

A copy of the GNU General Public License can be found at:

<http://www.gnu.org/licenses/gpl.html>

or you can write to:

Free Software Foundation, Inc.
59 Temple Place – Suite 330
Boston, MA 02111-1307
USA.

*/

```
void usage(char* name) {
    printf(
        "_____ \n"
        "\n"
        ",sSSs, Beta v1.0.\n"
        " dS\ " dP Multi-format shellcode encoding tool.\n"
        ".SP dSS\ " Copyright (C) 2003 by Berend-Jan Wever\n"
        " dS' Sb <skylined@edup.tudelft.nl>\n"
        ".SP dSSP' Encodes shellcode to a variety of formats.\n"
        "_ iS:_____ \n"
        "\n"
        "Usage: %s option [#chars]\n"
        "Options:\n"
        "--help Display this information.\n"
        "--length Print the length of the input.\n"
        "--check Check the input for 0x0, 0xA and 0xD
characters.\n"
        "--string Encode as a string for C.\n"
        "\"\xAA\xBB\xCC...\"\n"
        "--chars Encode as an array of chars for C.\n"
        "\xAA', '\xBB', '\xCC', ...'\n"
        "--hex Encode as an array of hex values for
C.\n"
        "0xAA, 0xBB, 0xCC, ...'\n"
        "--amp Encode as entities for HTML.\n"
        "&#AA;&#BB;&#CC;...\n"
        "--utf-8 Encode using utf-8.\n"
        "%AA%BB%CC...\n"
        "--utf-16 Encode using utf-16.\n"
        "%uBBAA%uDDCC...\n"
        "#chars Optional number of encoded characters to
print\n"
```

```

" after which a newline gets printed.\n"
"\n"
"Input is read from stdin, the result is written to stdout. When
encoding into\n"
"words and the input has an uneven number of bytes, the input is
padded with a\n"
"NOP (0x90) byte.\n",
    name
);
}

// used for encoding each byte by itself
void encoder_per_byte(int chars_per_line, char* line_header, char*
line_footer,
    char* byte_header, char* byte_format, char* byte_footer,
char* byte_separator) {
    int input=0, count=0;
    // line header and footer only printed when we have a max. chars per
line.
    if (chars_per_line>0) printf("%s", line_header);
    // read 1 byte input from stdin
    while ((input = getchar()) != EOF) {
        // if we've allready printed chars we might have to print seperators
        if (count > 0) {
            // we have to seperate bytes from each other with this:
            printf("%s", byte_separator);
            // if we've allready printed enough chars on this line, end it &
start a new one:
            if (chars_per_line>0 && count % chars_per_line == 0)
                printf("%s%s", line_footer, line_header);
        }
        // print the byte (with it's own header and footer) and count it.
        printf("%s", byte_header);
        printf(byte_format, input);
        printf("%s", byte_footer);
        count++;
    }
    // line header and footer only printed when we have a max. chars per
line.
    if (chars_per_line>0) printf("%s", line_footer);
}

// used for encoding two bytes into a little endian word (AA, BB -> BBAA)
void encoder_per_word(int chars_per_line, char* line_header, char*
line_footer,
    char* word_header, char* word_footer, char* word_separator) {
    int input1=0, input2=0, count=0;
    // line header and footer only printed when we have a max. chars per
line.
    if (chars_per_line>0) printf("%s", line_header);
    // read 1 byte input from stdin

```

Securiteam: [TOOL] Beta – Multi-Format Shellcode Encoding Tool

```
while ((input1 = getchar()) != EOF) {
// read another byte input from stdin
    input2 = getchar();
// if number of input bytes=uneven pad with 0x90
    if (input2 == EOF) input2 = 0x90;
// if we've allready printed chars we might have to print seperators
    if (count > 0) {
        // we have to seperate words from each other with this:
        printf("%s", word_separator);
        // if we've allready printed enough chars on this line, end it &
start a new one:
        if (chars_per_line>0 && count % chars_per_line == 0)
            printf("%s%s", line_footer, line_header);
    }
    // print the word (with it's own header and footer) and count the
bytes.
    printf("%s%02x%02x%s", word_header, input2, input1, word_footer);
    count+=2;
}
// line header and footer only printed when we have a max. chars per
line.
if (chars_per_line>0) printf("%s", line_footer);
}
```

```
int main(int argc, char* argv[]) {
    int chars_per_line = (argc>2 ? atoi(argv[2]) : -1);
    int i=0, j=0, error=0;
    if (chars_per_line == 0) {
        printf("Illegal number of chars per line. Type \"%s --help\" for
help.\n", argv[0]);
        error=1;
    } else if (argc<2 || strcmp(argv[1], "--help")==0) {
        // display usage information
        usage(argv[0]);
    } else if (strcmp(argv[1], "--length")==0) {
        // output length of input
        while (getchar()!=EOF) i++;
        printf("%d\n", i);
    } else if (strcmp(argv[1], "--check")==0) {
        // check for 0x0, 0xA and 0xD
        while ((j=getchar())!=EOF) {
            i++;
            if (j==0x0 || j==0xA || j==0xD) {
                printf("Character %d is 0x%02x!\n", i, j);
                error = 1;
            }
        }
    }
    if (!error)
        printf("Shellcode is NULL, CR and LF free.\n");
    } else if (strcmp(argv[1], "--string")==0) {
        // dump "\xAA\xBB\xCC..." encoded string.
    }
```

Securiteam: [TOOL] Beta – Multi-Format Shellcode Encoding Tool

```
encoder_per_byte(chars_per_line, " \\", "\\n", "\\x", "%02x", "",
");
} else if (strcasecmp(argv[1], "--chars")==0) {
// dump '\xAA', '\xBB', '\xCC', ... encoded chars
encoder_per_byte(chars_per_line, " ", "\n", "\\x", "%02x", "", "
");
} else if (strcasecmp(argv[1], "--hex")==0) {
// dump 0xAA, 0xBB, 0xCC, ... encoded bytes
encoder_per_byte(chars_per_line, " ", "\n", "0x", "%02x", "", " ");
} else if (strcasecmp(argv[1], "--amp")==0) {
// dump '&#aa;&#bbb;&#c;...' encoded string
encoder_per_byte(chars_per_line, " \\", "\\n", "&#", "%d", ";", "");
} else if (strcasecmp(argv[1], "--utf-8")==0) {
// dump "%AA%BB%CC..." encoded string
encoder_per_byte(chars_per_line, " \\", "\\n", "%", "%02x", "", "");
} else if (strcasecmp(argv[1], "--utf-16")==0) {
// dump "%uBBAA%uDDCC..." encoded string
encoder_per_word(chars_per_line, " \\", "\\n", "%u", "", "");
} else {
printf("Unknown option. Type \"%s --help\" for help.\n", argv[0]);
error=1;
}

return (error ? EXIT_FAILURE : EXIT_SUCCESS);
}
```

ADDITIONAL INFORMATION

To keep updated with the tool visit the project's homepage at:

<http://www.edup.tudelft.nl/~bjwever/src/beta.c>

<http://www.edup.tudelft.nl/~bjwever/src/beta.c>

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====

=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.