개요

system()함수 개요

system()은 PHPoC 시스템 관련 명령 실행 기능을 제공하는 내장 함수입니다.

명령어는 예고없이 추가 및 삭제될 수 있습니다.

명령 형식

시스템 함수는 다음과 같은 형식으로 되어 있습니다.

```
string system(string $command_string[, string arg1, string arg2, ...]);
```

system()은 string 타입의 인수를 받아서 해당 명령을 처리한 후 결과값을 string 형태로 반환합니다.

형식 1: 인수없이 명령어 문자열만 사용

다음은 인수없이 명령어 문자열로만 구성된 system() 함수 사용예 입니다.

```
<?php
system("php main.php");  // Run main.php
?>
```

```
<?php
system("php -d 3 main.php");  // Run main.php (restart delay: 3 seconds)
?>
```

```
<?php
// Run main.php (CPU time: 500us, restart delay: 3 seconds)
system("php -t 500 -d 3 main.php");
?>
```

형식 2: 인수와 함께 명령 문자열 사용

%1, %2와 같이 %뒤에 숫자가 나오는 형식의 문구에 인수가 차례로 치환되는 형식입니다. Space나 Control Character가 포함된 인자들이 사용될 때 적합한 형식입니다. 다음은 인수와 함께 명령 문자열을 사용한 system 함수 사용예 입니다.

```
<?php
$script = "main.php";
system("php %1",$script);  // Run main.php
?>
```

```
<?php
$delay = "3";
$script = "main.php";
```

```
system("php -d %1 %2", $delay, $script);  // Run main.php (restart delay: 3 seconds)
?>
```

```
<?php
$php_id = "0";
$cpu_time = "500";
$delay = "3";
$script = "main.php";
// Run main.php (CPU time: 500us, restart delay: 3 seconds)
system("php -t %2 -d %3 %4", $php_id, $cpu_time, $delay, $script);
?>
```

PHPoC 제어 및 정보 명령

uname 명령

PHPoC 버전, 프로세서 정보, 하드웨어 정보등 시스템 정보를 반환합니다. 함수의 인수로 "-"로 시작되는 스트링이 필요하며, 하나의 스트링으로 된 여러 개의 인수도 지원합니다.
예: system("uname -sv");

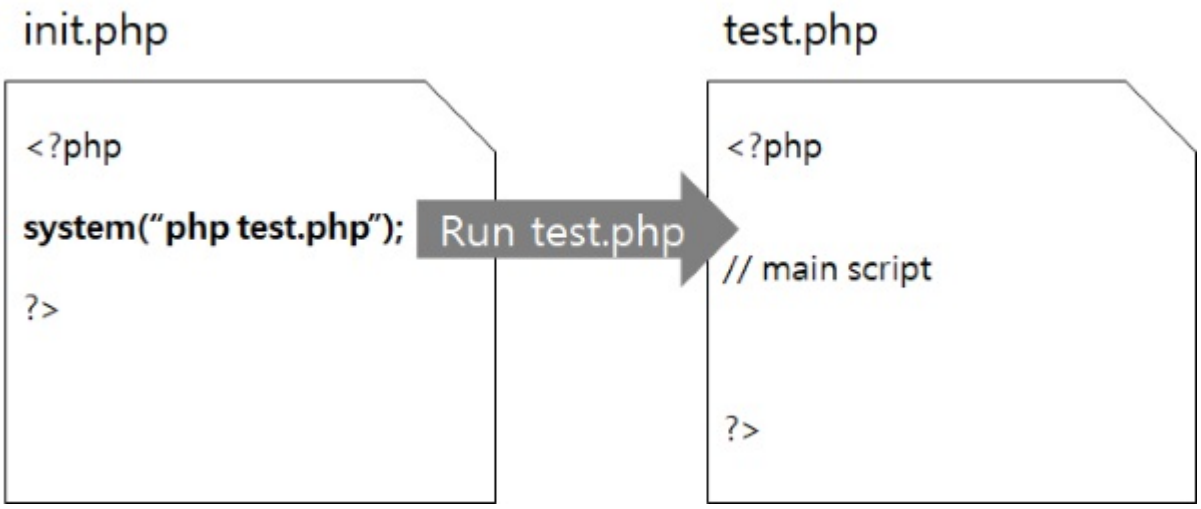
인수	설명
s	PHPoC 엔진 이름
v	PHPoC 엔진 버전
p	프로세서 정보
i	하드웨어 플랫폼 정보

다음은 uname 명령 사용 예입니다.

```
<?php
// when the product is PBH-101
echo system("uname -s"), "₩r₩n";  // OUTPUT: PHPoC
echo system("uname -v"), "₩r₩n";  // OUTPUT: 1.0.0
echo system("uname -i"), "₩r₩n";  // OUTPUT: PBH-101
echo system("uname -p"), "₩r₩n";  // STM32F407 Cortex-M4F 168MHz
echo system("uname -svpi"), "₩r₩n";
// OUTPUT: PHPoC 1.0.0 STM32F407 Cortex-M4F 168MHz PBH-101
?>
```

php 명령

제품이 기동되면 최초로 실행되는 스크립트 파일은 init.php입니다. init.php 파일에 사용자가 원하는 기능을 프로그램 해도 되지만, php 시스템 명령으로 다른 스크립트 파일을 실행시킬 수 있습니다. 또한 이 명령으로 스크립트 태스크의 CPU 점유율을 조절할 수 있습니다



php 명령에 형식은 다음과 같습니다.

- string system("php [-t $cpu_time -d $restart_delay $script_name]");

이 명령은 실행시키는 스크립트 이름을 반환합니다.

인수	설명
$cpu_time	PHPoC 엔진에서 루프당 스크립트를 실행하는 시간입니다. 이 항목이 생략되면 기본값은 500us입니다. (설정가능값: 10 ~ 10000) 이 값이 크면 클수록 해당 스크립트의 CPU 점유율이 높아집니다. 스크립트 CPU 점유율이 높아지면 네트워크 부하가 많은 경우 패킷 유실이 발생 할 수 있습니다.
$restart_delay	PHPoC 스크립트 실행이 종료되면 이 항목에서 설정한 시간(초)후에 스크립트가 자동으로 재시작됩니다. 만약 이 값이 0 이면 스크립트는 재시작되지 않습니다. 생략되면 기본값은 5 입니다. 디버거 연결시 자동으로 재시작되지 않습니다.
$script_name	$script_name이 인수로 사용되면 현재 실행하고 있는 스크립트가 종료후 다음에 실행할 스크립트 파일 이름을 지정합니다. 이 파라미터가 생략되면 현재의 태스크의 CPU 점유율을 즉시 설정합니다.

다음은 현재 스크립트가 종료되면 test.php 스크립트 파일이 실행되도록 하는 코드입니다.

- system("php test.php"); // run test.php after finishing this script

reboot 명령

reboot 명령으로 PHPoC 스크립트 및 시스템을 다시 시작할 수 있습니다. 형식은 다음과 같습니다.

- string system("reboot %1 [ms]", $target);

reboot 명령은 $target에 따라서 다음과 같이 동작합니다.

$target	설명
php	PHPoC 엔진을 재시작합니다.
sys	시스템(제품)을 재시작합니다.

[ms]는 PHPoC 스크립트 재시작 대기 시간이고 단위는 ms입니다. [ms]는 생략될 수 있으며 생략 시 100ms입니다. 이 명령의 반환값은 기다리는 시간 [ms] 입니다.

```
<?php
echo "Restart PHPoC in 1 second...₩r₩n";
system("reboot php 1000");

while(1);
?>
```

플래시 메모리 명령

환경변수 영역

제품이 동작할 때 필요한 정보 중에 전원이 꺼져도 보관되어야 할 정보가 있습니다. 이런 정보들은 비휘발성(non-volatile) 메모리인 플래시 메모리에 저장하면 됩니다.

PHPoC 플래시 메모리에는 다음과 같이 시스템에서 사용하는 시스템 데이터 영역과 사용자가 사용 가능한 사용자 데이터 영역이 있습니다.

분류	설명	데이터 종류
시스템 데이터 영역 (/mmap/envs)	시스템이 동작 중에 사용하는 값을 저장하는 공간	네트워크 관련 데이터 기타 시스템 데이터
사용자 데이터 영역 (/mmap/envu)	사용자 데이터를 저장하는 공간	사용자 데이터

이 문서에서는 사용자 데이터 저장공간에 관련한 명령 정보만 제공됩니다. 시스템 데이터 저장공간에 관련하여서는 다른 문서를 참고해 주시기 바랍니다.

nvm 명령

플래시 메모리 영역에 데이터를 저장하려면 "nvm wkey" 명령으로 플래시 저장용 인증키를 생성한 후 "nvm write" 명령으로 플래시에 저장합니다.

- system("nvm wkey %1", $target);

플래시 저장용 인증키를 생성하고 "nvm write" 명령의 인수로 사용될 키값을 반환합니다.

인수	설명
$target	저장할 영역을 지정 (envs: 시스템 데이터 영역, envu: 사용자 데이터 영역)

- system("nvm write envs/envu wkey env");

플래시에 저장을 합니다.

인수	설명
envs/envu	envs - 시스템 데이터 영역 envu - 사용자데이터영역
wkey	"nvm wkey" 명령에서 생성된 인증키
env	저장할 데이터

플래시 메모리에 저장 후 2초 이내에 동일한 영역에 저장할 수 없습니다. 또한 플래시 메모리는 최대 저장 횟수가 제한되어 있으니 반복적인 저장용도로 사용시 주의가 필요합니다.

다음은 "abcdefghij"를 플래시 메모리의 사용자 데이터 영역에 저장하는 예제입니다.

```
<?php
$str = "abcdefghij";

echo "setup /mmap/envu (user non-volatile meory)₩r₩n";
$wkey = system("nvm wkey envu");
echo "write ₩$str to /mmap/envu₩r₩n";
system("nvm write envu $wkey %1", $str);  // write $str  to /mmap/envu (flash)

echo "open /mmap/envu and read it₩r₩n";
$pid_envu = pid_open("/mmap/envu");  // open /mmap/envu
$buf = "";
pid_read($pid_envu, $buf, 10);  // read /mmap/envu
echo "/mmap/envu : $buf₩r₩n";

while(1);
?>
```

암호화 명령

데이터 암호화/복호화 (Encryption/Decryption)

rc4 명령

RC4는 Ron Rivest 가 만든 스트림 암호(stream cipher)로서 TLS 및 WEP에 사용되고 있습니다. RC4는 symmetric 방식의 암호화 알고리즘이므로 암호화와 복호화는 모두 같은 명령을 사용합니다.

다음은 PHPoC에서 RC4 암호화/복호화 방법입니다.

- system("rc4 init %1", $rc4_key);

RC4 암호와/복호화를 위해서 암호 엔진을 초기화 하는 함수입니다. 암호화/복호화 전에 반드시 이 함수를 실행해야 합니다. 이 명령은 암호화/복호화 할 때 사용되는 context가 반환됩니다.

인수	설명
$rc4_key	RC4암호화를 위한 미리 정해진 키

- system("rc4 crypt %1 %2", $rc4, $rc4_text);

RC4 암호화/복호화 작업을 합니다. $rc4는 초기화 할 때 만들어진 context입니다. $rc4_text는 암호화 할 대상인 평문(plain text) 혹은 복호화할 암호문(cipher)입니다. 암호화 혹은 복호화된 데이터가 반환됩니다.

인수	설명
$rc4	초기화 할 때 만들어진 context
$rc4_text	암호화 할 대상인 평문(plain text) 혹은 복호화할 암호문(cipher)

- system("rc4 skip %1", $rc4);

RC4 암호화/복호화 건너뛸 때 사용 합니다. RC4 취약성 개선을 위해서는 skip 과정을 반드시 수행해야 합니다.

인수	설명
$rc4	초기화 할 때 만들어진 context

다음은 암호화/복호화 예제입니다.

```
// encryption
$rc4 = system("rc4 init %1", $rc4_key);  // initialize
$out = system("rc4 crypt %1 %2", $rc4, $rc4_pt);  // encryption

// decryption test
$rc4 = system("rc4 init %1", $rc4_key);  // initialize
$out = system("rc4 crypt %1 %2", $rc4, $rc4_ct);  // decryption
```

des 명령

DES(Data Encryption Standard)는 symmetric-key 알고리즘을 사용하는 데이터 암호화 방법입니다. DES는 전수검사를 통해서 완전히 해독 되었기 때문에 민감한 정보 암호화 용으로는 적합하지 않습니다. 3번의 DES 암호화/복호화 과정을 수행하는 triple DES는 개선된 암호화 강도를 제공합니다. DES 암호화 방식은 ECB와 CBC 2가지 방법이 있습니다.

다음은 ECB 방식의 암호화/복호화 함수입니다.

- system("des init ecb/ede3_ecb enc/dec %1", $ecb_key);

DES 엔진을 초기화 하는 함수입니다. 암호화/복호화에 사용될 context를 반환합니다.

인수	설명
ecb/ede3_ecb	ecb - DES ECB ede3_ecb - 3DES ECB
enc/dec	enc - 암호화 dec - 복호화
$ecb_key	ECB 암호화/복호화에 사용되는 64비트 키

- system("des crypt %1 %2", $des, $text);

전에 실행된 초기화 함수의 설정에 따라서 암호화/복호화를 수행합니다. 암호화/복호화된 암호문 혹은 평문을 반환합니다.

인수	설명
$des	초기화시 반환받은 context
$text	암호화/복호화할 평문 혹은 암호문

다음은 ECB방식의 DES 암호화/복호화 예제입니다.

```
// encryption
$des = system("des init ecb enc %1", $ecb_key);  // initialize
$out = system("des crypt %1 %2", $des, $ecb_pt); // encryption

// decryption
$des = system("des init ecb dec %1", $ecb_key);  // initialize
$out = system("des crypt %1 %2", $des, $ecb_ct); // decryption
```

다음은 ECB방식의 Triple DES 암호화/복호화 예제입니다.

```
// encryption
$des = system("des init ede3_ecb enc %1", $ecb_key);
$out = system("des crypt %1 %2", $des, $ecb_pt);

// decryption
$des = system("des init ede3_ecb dec %1", $ecb_key);
```

```
$out = system("des crypt %1 %2", $des, $ecb_ct);
```

다음은 CBC 방식의 암호화/복호화 함수입니다.

- system("des init cbc/ede3_cbc enc/dec %1 %2", $cbc_key, $iv);

DES 엔진을 초기화 하는 함수입니다. 암호화/복호화에 사용될 context를 반환합니다.

인수	설명
cbc/ede3_cbc	cbc - DES CBC ede3_cbc - 3DES CBC
enc/dec	enc - 암호화 dec - 복호화
$cbc_key	CBC 암호화/복호화에 사용되는 64비트 키
$iv	64비트 initialization vector

- system("des crypt %1 %2", $des, $text");

전에 실행된 초기화 함수에 따라서 암호화/복호화를 수행합니다. 암호화/복호화된 암호문 혹은 평문을 반환합니다.

인수	설명
$des	초기화시 반환받은 context
$text	암호화/복호화할 평문 혹은 암호문

다음은 CBC방식의 DES 암호화/복호화 예제입니다.

```
// encryption
$des = system("des init cbc enc %1 %2", $cbc_key, $cbc_iv);  // initialize
$out = system("des crypt %1 %2", $des, $cbc_pt);             // encryption

// decryption
$des = system("des init cbc dec %1 %2", $cbc_key, $cbc_iv);  // initialize
$out = system("des crypt %1 %2", $des, $cbc_ct);             // decryption
```

aes 명령

The AES(Advanced Encryption Standard)는 미국 NIST에서 만든 데이터 암호화 스펙으로 DES를 대신하는 표준으로 사용되고 있습니다. AES는 암호화 방식은 ECB와 CBC 2가지 방법이 있습니다.

다음은 ECB 방식의 암호화/복호화 방법에 대한 설명입입니다.

- system("aes init ecb enc/dec %1", $ecb_key);

AES 엔진을 초기화 하는 함수입니다. 암호화/복호화시 사용될 context를 반환합니다.

인수	설명
enc/dec	enc - 암호화 dec - 복호화
$ecb_key	ECB 암호화/복호화에 사용되는 128/192/256 비트 키

- system("aes crypt %1 %2", $aes, $text);

전에 실행된 초기화 함수의 설정에 따라서 암호화/복호화를 수행합니다. 암호화/복호화된 암호문 혹은 평문을 반환합니다.

인수	설명
$aes	초기화시 반환받은 context
$text	암호화/복호화할 평문 혹은 암호문

다음은 CBC 방식의 암호화/복호화 함수입니다.

- system("aes init cbc enc/dec %1 %2", $cbc_key, $iv);

AES 엔진을 초기화 하는 함수입니다. 암호화/복호화시 사용될 context를 반환합니다.

인수	설명
$enc/dec	enc - 암호화 dec - 복호화
$cbc_key	CBC 암호화/복호화에 사용되는 128/192/256 비트 키
$iv	128비트 initialization vector

- system("aes crypt %1 %2", $aes, $text);

전에 실행된 초기화 함수에 따라서 암호화/복호화를 수행합니다. 암호화/복호화된 암호문 혹은 평문을 반환합니다.

인수	설명
$aes	초기화시 반환받은 context
$text	암호화/복호화할 평문 혹은 암호문

다음은 CBC방식의 AES 암호화/복호화 예제입니다.

```
// encryption
$aes = system("aes init cbc enc %1 %2", $cbc_key, $cbc_iv);
$out = system("aes crypt %1 %2", $aes, $cbc_pt16);

// decryption
$aes = system("aes init cbc dec %1 %2", $cbc_key, $cbc_iv);
$out = system("aes crypt %1 %2", $aes, $cbc_ct16);
```

seed 명령

SEED는 한국인터넷진흥원에서 개발한 암호화 알고리즘입니다. SEED는 암호화 방식은 ECB와 CBC 2가지 방법이 있습니다.

다음은 ECB 방식의 암호화/복호화 함수입니다.

- system("seed init ecb enc/dec %1", $type, $ecb_key);

SEED 엔진을 초기화 하는 함수입니다. 암호화/복호화시 사용될 context를 반환합니다.

인수	설명
enc/dec	enc - 암호화 dec - 복호화
$ecb_key	ECB 암호화/복호화에 사용되는 128/256 비트 키

- system("seed crypt %1 %2", $seed, $text);

전에 실행된 초기화 함수의 설정에 따라서 암호화/복호화를 수행합니다. 암호화/복호화된 암호문 혹은 평문을 반환합니다.

인수	설명
$seed	초기화시 반환받은 context
$text	암호화/복호화할 평문 혹은 암호문

다음은 CBC 방식의 암호화/복호화 함수입니다.

- system("seed init cbc enc/dec %1 %2", $cbc_key, $iv);

SEED 엔진을 초기화 하는 함수입니다. context를 반환합니다.

인수	설명
$enc/dec	enc - 암호화 dec - 복호화
$cbc_key	CBC 암호화/복호화에 사용되는 128/256 비트 키
$iv	128비트 initialization vector

- system("seed crypt %1 %2", $seed, $text);

전에 실행된 초기화 함수의 설정에 따라서 암호화/복호화를 수행합니다.암호화/복호화된 암호문 혹은 평문을 반환합니다.

인수	설명
$seed	초기화시 반환받은 context
$text	암호화/복호화할 평문 혹은 암호문

다음은 CBC방식의 AES 암호화/복호화 예제입니다.

```
// encryption
$seed = system("seed init cbc enc %1 %2", $cbc_key, $cbc_iv);
$out = system("seed crypt %1 %2", $seed, $cbc_pt32);

// decryption
$seed = system("seed init cbc dec %1 %2", $cbc_key, $cbc_iv);
$out = system("seed crypt %1 %2", $seed, $cbc_ct32);
```

base64 명령

BASE64는 binary 데이터를 ASCII 데이터로 변환하는 알고리즘으로 email과 XML에서 사용됩니다.
BASE64는 용도에 따라서 여러 변형이 있는데 PHPoC에서는 표준형, URL형, MIME형을 지원합니다.

다음은 BASE64 암호화/복호화 함수입니다.

- system("base64 enc/dec %1 [std/mime/url]", $msg);

인수	설명
enc/dec	enc - 암호화 dec - 복호화
$msg	암호화/복호화할 평문 혹은 암호문
std/mime/url	std - 표준형 mime - MIME url - URL 생략하면 표준형으로 동작합니다.

다음은 BASE64의 예제입니다.

```
$enc_out = system("base64 enc %1", $msg0);
$dec_out = system("base64 dec %1", $enc_out);
```

해시(Hash) 명령

crc 명령

crc 명령은 8/16/32비트 CRC를 계산하는 시스템 명령이며 형식은 다음과 같습니다.

- system("crc bits %1 [init div msb/lsb]", $msg);

이 명령의 반환값은 계산된 CRC 값 입니다.

인수	설명
bits	8 - 8bit CRC 계산 16 - 16bit CRC 계산 32 - 32bit CRC 계산
$msg	CRC를 계산할 원본 데이터
init	CRC 초기값 이 항목을 생략하면 기본값을 8비트-ff, 16비트-1d0f, 32비트-ffffffff 입니다.
div	CRC계산할 때 사용되는 divisor(polynomial)입니다. 이 항목을 생략하면 기본값은 8비트: e0 16비트: 1021 32비트: edb88320 입니다.
msb/lsb	CRC 계산 순서입니다. msb는 byte 데이터의 상위->하위 비트 순서로 계산되고, lsb는 하위->상위 비트 순서로 계산됩니다. 생략시 기본값: 8비트 - lsb, 16비트 - msb, 32비트 - lsb

다음은 각 CRC종류마다 CRC를 계산하는 예제코드입니다.

```
<?php

$string = "123456789";

printf("CRC-16-ANSI : %04x₩r₩n", (int)system("crc 16 %1 0000 a001 lsb", $string));

printf("CRC-16-Modbus : %04x₩r₩n", (int)system("crc 16 %1 ffff a001 lsb", $string));

printf("CRC-CCITT FFFF: %04x₩r₩n", (int)system("crc 16 %1 ffff 1021 msb", $string));

printf("CRC-CCITT 1D0F: %04x₩r₩n", (int)system("crc 16 %1 1d0f 1021 msb", $string));

printf("CRC-CCITT XModem : %04x₩r₩n", (int)system("crc 16 %1 0000 1021 msb", $string));

$crc16_out = (int)system("crc 16 123456789 %1 8408 lsb", $string);
$crc16_out = bin2int(int2bin($crc16_out, 2, true), 0, 2);
printf("CRC-CCITT Kermit : %04x₩r₩n", $crc16_out);

$crc16_out = (int)system("crc 16 123456789 ffff 8408 lsb");
$crc16_out = $crc16_out ^ 0xffff;
printf("CRC-CCITT PPP    : %04x₩r₩n", $crc16_out);

$crc16_out = ~(int)system("crc 16 %1 0000 a6bc lsb", $string);
$crc16_out = bin2int(int2bin($crc16_out, 2, true), 0, 2);
printf("CRC-16-DNP       : %04x₩r₩n", $crc16_out);
```

```
?>
```